

End-to-End Quantum Algorithms for the Jones Polynomial

Tuomas Laakkonen^{1,*}, Enrico Rinaldi¹, Chris N. Self¹, Eli Chertkov², Matthew DeCross²,
David Hayes², Brian Neyenhuis², Marcello Benedetti¹, and Konstantinos Meichanetzidis^{1,†}

¹*Quantinuum, Partnership House, Carlisle Place, London SW1P 1BX, United Kingdom*

²*Quantinuum, 303 S. Technology Ct., Broomfield, Colorado 80021, USA*



(Received 11 April 2025; revised 2 February 2026; accepted 19 February 2026; published 9 June 2026)

We present an end-to-end algorithmic pipeline where a noisy digital quantum computer is used to approximate the value of the Jones polynomial at the fifth root of unity for any input link, i.e., a closed braid. This problem is DQC1-complete for Markov-closed braids and BQP-complete for Plat-closed braids, and we accommodate both versions of the problem. Even though it is widely believed that DQC1 is strictly contained in BQP, and so is “less quantum,” the resource requirements of classical algorithms for the DQC1 version are at least as high as for the BQP version, and so we potentially gain “more advantage” by focusing on Markov-closed braids in our exposition. We demonstrate our quantum algorithm on Quantinuum’s H2-2 quantum computer and show the effect of problem-tailored error-mitigation techniques. Further, leveraging that the Jones polynomial is a link invariant, we construct an efficiently verifiable benchmark to characterize the effect of noise present in a given quantum processor. In parallel, we develop and benchmark the state-of-the-art tensor-network-based classical algorithms for computing the Jones polynomial. The reconfigurable tools provided in this work allow for precise resource estimation to identify minimum link sizes for near-term quantum advantage in practice for a meaningful quantum-native problem in knot theory, if a candidate set of links are provided.

DOI: [10.1103/jgv8-l3j1](https://doi.org/10.1103/jgv8-l3j1)

I. INTRODUCTION

Knot theory offers a wide range of computational problems that are notably difficult to solve. A paradigmatic example is the evaluation of certain *link invariants*, quantities that do not change for knots that are equivalent in the topological sense. One such quantity is the Jones polynomial, a Laurent polynomial with coefficients and powers determined solely by the link topology. This means that if two links have different Jones polynomials, then the links are topologically inequivalent [1]. The exact evaluation of the Jones polynomial is in general #P-hard [2], meaning that both classical and quantum algorithms are believed to be inefficient at this task.

We instead consider the problem of additively approximating the Jones polynomial, up to an exponentially large-scale factor, at nonlattice roots of unity for a *given link*. Specifically, the problem we study is defined in terms of

braids, as any link can be formed by “closing” a braid. Remarkably, braids have some unitary representations, and there are two ways to close a braid. For the “Markov closure,” the problem reduces to estimating the weighted trace of a unitary matrix, and is DQC1-complete [3]. For the “Plat closure,” the problem reduces to estimating a quantum amplitude, i.e., an element of a unitary matrix, and is BQP-complete [4]. We illustrate these in the top and bottom panels of Fig. 1(a). It is widely believed that problems in these complexity classes cannot be efficiently solved by a classical computer in the worst case. Thus, they are excellent candidates for demonstrating a separation between classical and quantum models of computation.

The DQC1-complete formulation of the problem can be solved in the restricted model of quantum computation called “one clean qubit.” There, a single qubit is initialized in a pure state, while the remaining ones are in the maximally mixed state, and measurements are performed only at the end of the computation [5]. In this work, we instead use a universal *gate-based digital quantum computer*, i.e., a BQP machine, and provide an end-to-end quantum algorithm that can treat both DQC1 and BQP versions of the problem. We choose the fifth root of unity as the evaluation point of the Jones polynomial. This is a well-studied case for which we can use the Fibonacci unitary representation of braids [3,6]. Notably, it is approximately

*Contact author: tsrl@mit.edu

†Contact author: k.mei@quantinuum.com

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article’s title, journal citation, and DOI.

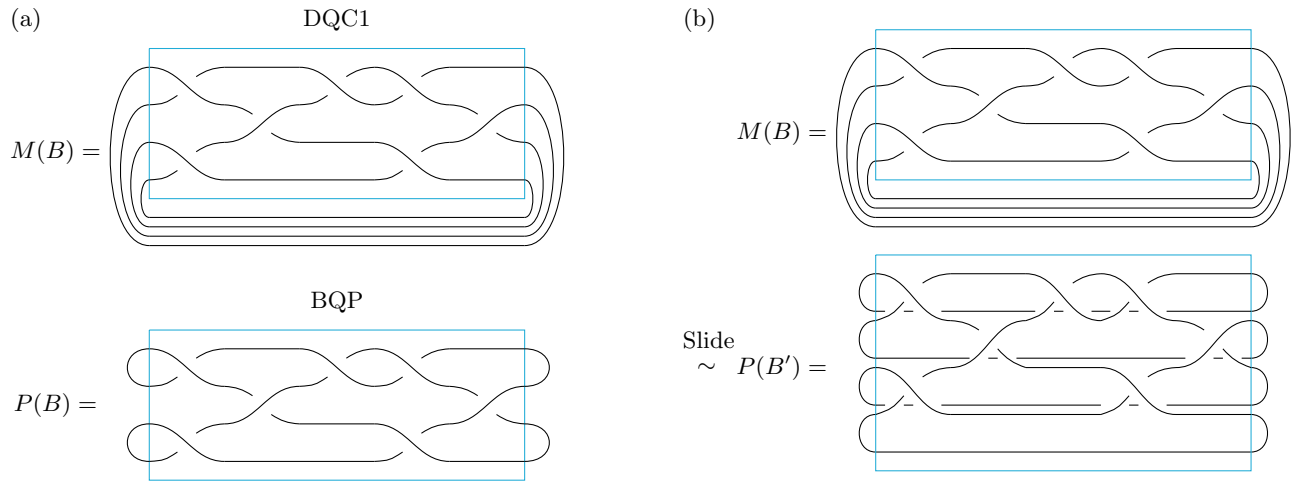


FIG. 1. (a) Markov closure $M(B)$ and Plat closure $P(B)$ of braid B , which we enclose in the blue box. Evaluating $V_{M(B)}$ to additive error is DQC1-complete and evaluating $V_{P(B)}$ to additive error is BQP-complete. In general $M(B)$ and $P(B)$ are topologically inequivalent. (b) Using slide moves, the Markov closure $M(B)$ can be reduced to the Plat closure $P(B')$ of a braid B' on twice the number of strands and more crossings in general.

universal for quantum computation [7]. Given this unitary representation of braids, we focus on estimating the desired weighted trace or amplitude, employing the Hadamard test as the key algorithmic primitive. Using its control-free variant, and making use of echo verification [8] and efficient compilation of braid unitaries, we minimize the circuit depth and number of shots, and therefore, the time to solution. We also employ methods for error detection [9] and error mitigation [10] to increase the accuracy and precision of the algorithm's estimate at the cost of minimal overhead in gates and shots.

Our highly optimized quantum algorithm can be used to define a problem-tailored benchmark for noisy digital quantum computers. The Jones polynomial, being a link invariant, remains unchanged when the link changes in a way that preserves its topology. Previous work leverages this property to design benchmarks and to determine the circuit sizes that a quantum processor can faithfully execute according to a given error budget and noise model. Ref. [11] employs the BQP version of the problem and chooses a lattice root of unity as the evaluation point for which the problem is actually in P, witnessed by the resulting circuits being Clifford which are classically efficiently simulable [12,13]. Ref. [14] uses the DQC1 version with the fifth root of unity as the evaluation point, albeit with a suboptimal compilation of the vanilla Hadamard test, to define a benchmark that cannot be efficiently verified were it to be scaled up. Our benchmark goes beyond previous proposals as it works for *both* the DQC1 and BQP versions and, importantly, is *efficiently verifiable*.

We begin by generating random braids whose Jones polynomial upon closure can be evaluated classically within a negligible amount of time. We then randomly

augment them to generate a set of topologically equivalent links, each of which has the same Jones polynomial as the original link. The corresponding circuits span various widths and depths and are guaranteed to ideally return the pre-computed answer up to efficient classical pre- and postprocessing. The circuits can finally be executed on a quantum computer of given specifications to characterize the effect of noise and identify the largest instances within our family of braids for which a desired accuracy can be achieved.

Having established an efficiently verifiable benchmark, we demonstrate how our pipeline can be used to estimate resource requirements for *quantum advantage*, in terms of *time to solution* and *energy consumption*. After fixing the output precision and assuming a *realistic noise model* for a *near-term* quantum computer of given specifications, we identify braid sizes that can be successfully attacked with our algorithm. We also quantify the classical computational resources necessary to compute the same quantity within the same error. Our family of random braids might neither be representative of the average case nor capture the most difficult instances. Yet, if we assume that the classically hard instances are at least as hard as our family of braids, then the benchmark provides a conservative estimate for where quantum advantage is possible.

Because the approximations of the Jones polynomial we consider are complete for DQC1 and BQP, which are classes widely believed to be classically hard, instances likely exist which are classically hard to solve. The tools provided in this work enable one to narrow down the focus of the search for instances exhibiting a near-term quantum computational advantage, specifically by lower bounding their *size*. We envision that this practical approach

inspires concrete quantification and identification of quantum advantage for other quantum-native problems. We hope our work stimulates new ideas toward quantum utility in the field of computational topology.

II. EVALUATING THE JONES POLYNOMIAL ON A NOISY DIGITAL QUANTUM COMPUTER

Any braid B can be composed from two generators, $\sigma_i^{\pm 1}$, shown in Fig. 2, by starting from the trivial braid on l strands, which we denote as 1_l . Any sequence of generators is called a *braid word*. The *Markov closure* $M(B)$ of a braid B is obtained by connecting the endpoints on one side with those on the other side and without adding any crossings. The *Plat closure* $P(B)$ is obtained by connecting neighboring pairs of strands with “cups and caps.” The two closures are shown in Fig. 1(a). Note that any Markov-closed braid can be deformed into a Plat-closed braid on twice the strands, as shown in Fig. 1(b). In Fig. 3 we show some examples of links represented as Markov-closed braids.

A. Problem definition

The DQC1-complete and BQP-complete problems of approximating the Jones polynomials $V_{M(B)}(t)$ and $V_{P(B)}(t)$ at $t = e^{i(2\pi/5)}$ of Markov-closed and Plat-closed braids B respectively, are stated as:

Problem. APPROX-JONES-MARKOV: Given a braid B on $n - 1$ strands and $c = O(\text{poly}(n))$ crossings, compute $R \in \mathbb{C}$ such that $|V_{M(B)}(e^{i(2\pi/5)}) - R| \leq \epsilon \phi^{n-2}$ with high probability, where $\epsilon^{-1} = O(\text{poly}(n))$.

Problem. APPROX-JONES-PLAT: Given a braid B on $n - 1$ strands and $c = O(\text{poly}(n))$ crossings, compute $R \in \mathbb{C}$ such that $|V_{P(B)}(e^{i(2\pi/5)}) - R| \leq \epsilon \phi^{(n/2)-\frac{3}{2}}$ with high probability, where $\epsilon^{-1} = O(\text{poly}(n))$.

Here $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. Note that the estimates are given up to an additive error with an exponentially large-scale factor of $\phi^{O(n)}$. This factor is necessary—in fact, [15, Theorem 1.2] shows that an additive approximation without this scaling factor is #P-hard to compute, and hence unlikely to be efficient for a quantum computer. Such additive bounds with an exponential scale factor are also common for similar quantum algorithms for approximate counting [16, Definition 1]. However, the completeness of these problems for DQC1 and BQP respectively indicates that even this weak approximation

$$\sigma_i = \begin{array}{c} i \\ \diagdown \quad \diagup \\ i+1 \end{array} \quad \sigma_i^{-1} = \begin{array}{c} i \\ \diagup \quad \diagdown \\ i+1 \end{array}$$

FIG. 2. Braid generators $\sigma_i^{\pm 1}$, out of which any braid B is composed.

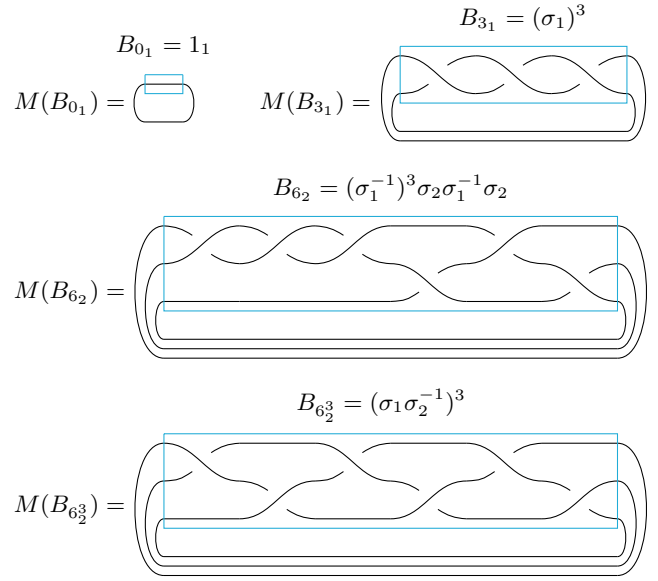


FIG. 3. Examples of Markov-closed braids $M(B)$ corresponding to the unknot 0_1 , the trefoil knot 3_1 , the 6_3 knot, and the 6_2^3 link. Each example braid is enclosed in a blue box and we also show its braid word.

of the Jones polynomial is likely to be hard for classical computers.

For the chosen evaluation point, the Fibonacci unitary representation of the braid generators, U_{σ_i} and $U_{\sigma_i^{-1}} = U_{\sigma_i}^\dagger = U_{\sigma_i}^*$, in the computational basis, is [6]:

$$\begin{aligned} \langle 101 | U_{\sigma_i} | 101 \rangle &= \phi^{-1} e^{i\frac{4\pi}{5}} & \langle 111 | U_{\sigma_i} | 101 \rangle &= \phi^{-\frac{1}{2}} e^{i\frac{7\pi}{5}} \\ \langle 101 | U_{\sigma_i} | 111 \rangle &= \phi^{-\frac{1}{2}} e^{i\frac{7\pi}{5}} & \langle 111 | U_{\sigma_i} | 111 \rangle &= -\phi^{-1} \\ \langle 110 | U_{\sigma_i} | 110 \rangle &= e^{i\frac{3\pi}{5}} & \langle 011 | U_{\sigma_i} | 011 \rangle &= e^{i\frac{3\pi}{5}} \\ \langle 010 | U_{\sigma_i} | 010 \rangle &= e^{-i\frac{4\pi}{5}} \end{aligned} \quad (1)$$

The submatrix defined by Eq. (1) is fully specified by unitarity and acts on the subspace of 3-bit strings that do not contain two consecutive zeros. Then, the braid-unitary U_B is obtained by composition of the 3-qubit gates $U_{\sigma_i^{\pm 1}}$ according to the braid word, such that a braid on $(n - 1)$ strands is represented as a quantum circuit on n qubits. As shown by the examples in Fig. 4, strands lie in between qubits such that U_{σ_i} acts on qubits $i - 1$, i , and $i + 1$. Therefore, U_B acts on a f_n -dimensional subspace of the n -qubit Hilbert space, where f_n is the n th Fibonacci number defined by the recurrence relation $f_{n+2} = f_{n+1} + f_n$, $f_0 = 0$, $f_1 = 1$, spanned by the Fibonacci basis states

$$\mathcal{F}_n = \{|s\rangle = \otimes_{i=0}^{n-1} |s_i\rangle \mid s_0 = 0 \wedge s_i + s_{i+1} > 0 \forall i\}. \quad (2)$$

In general, we call bit strings that contain no consecutive zeros (of which $\mathcal{F}_n$ is a subset) *Fibonacci strings*.

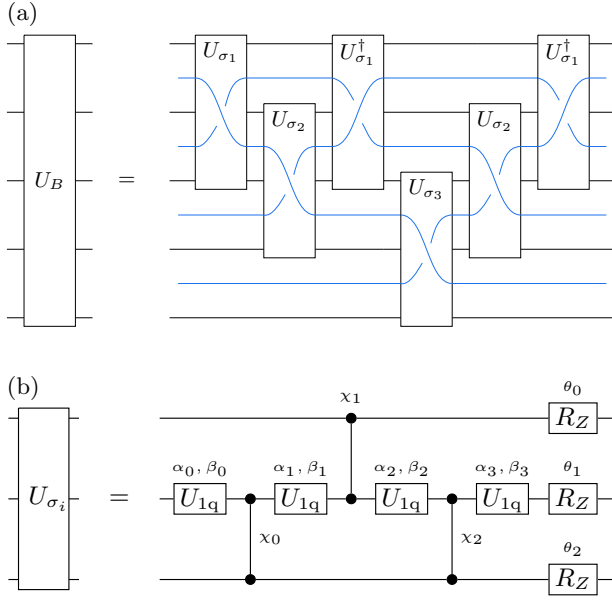


FIG. 4. Constructing quantum circuits for a braid using optimized circuits for the braid generators. (a) Example of the 5-qubit unitary for the 4-strand braid $B = \sigma_1 \sigma_2 \sigma_1^{-1} \sigma_3 \sigma_2 \sigma_1^{-1}$. The 3-qubit gates U_{σ_i} , from which any such braid-unitary is composed, are defined in Eq. (1). (b) Optimized circuit, composed of Z-phase gates $R_Z(\theta)$, native 1-qubit gates $U_{1q}(\alpha, \beta)$ and native 2-qubit (Ising) gates $R_{ZZ}(\chi)$ [17], implementing the 3-qubit gate $U_{\sigma_i}^k$ of Eq. (1). See Appendix D3 for the parameter values.

B. Quantum algorithms

Now we present the quantum algorithms that estimate $V_{M(B)}(e^{i(2\pi/5)})$ and $V_{P(B)}(e^{i(2\pi/5)})$, respectively, for a given braid B . It is possible to estimate $V_{M(B)}(e^{i(2\pi/5)})$ from the *weighted trace* of U_B as done by Shor and Jordan [3] in the one clean qubit model. In our work, equipped with a gate-based digital quantum computer, we simulate the one clean qubit model using the Hadamard test, as suggested by Aharonov, Jones, and Landau [4]. We express the desired quantity as

$$V_{M(B)}(e^{i\frac{2\pi}{5}}) = (-e^{-i\frac{3\pi}{5}})^{3w_B} \phi^{n-2} \mathbb{E}_{s \sim p(s)}[\langle s | U_B | s \rangle], \quad (3)$$

where $w_B = \sum_{\sigma \in B} \text{sign}(\sigma)$ is the writhe of the braid defined as the difference between positive and negative crossings, and the basis states are sampled from the following probability distribution (see Appendix B):

$$p(s) = \begin{cases} \frac{\phi^{s_{n-1}}}{\phi^{n-1}} & \text{if } |s\rangle \in \mathcal{F}_n \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

To estimate the quantity in Eq. (3), we adapt the control-free echo-verification protocol cf-ev [8]. This protocol can be understood as a control-free version of the Hadamard test [18], which also combines elements of the SWAP test [19,20] to reduce the variance of the vanilla Hadamard test

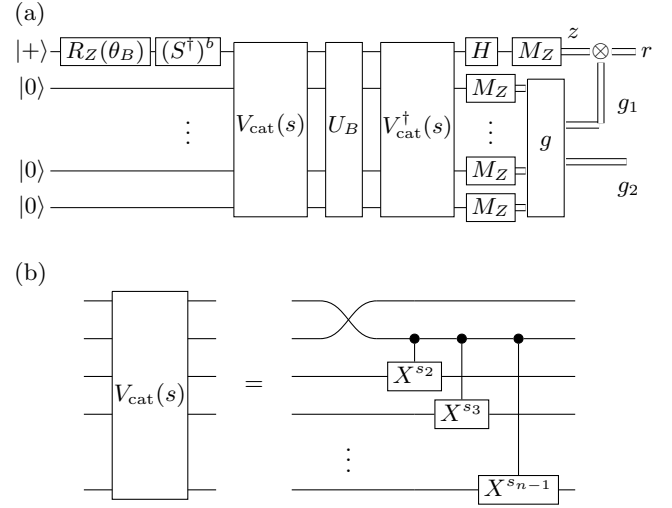


FIG. 5. The control-free echo-verification Hadamard test. (a) We employ cat states prepared by V_{cat} . We measure all qubits and postprocess with a function g that returns two outputs g_1 and g_2 . Sampling this circuit multiple times allows us to estimate both $\mathbb{E}[\text{Re}(s|U_B|s)]$ for $b=0$ and $\mathbb{E}[\text{Im}(s|U_B|s)]$ for $b=1$ using the g_1 output, up to additive error [see Fig. 4(a) for an example of a braid-unitary U_B]. The g_2 output can be used to detect when hardware errors have occurred. (b) Circuit preparing the n -qubit cat state $\frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + |s\rangle) = V_{\text{cat}}(s)(|+\rangle \otimes |0\rangle^{\otimes n-1})$.

[4] (see Appendix A for further details). Given U , $|\psi\rangle$, and an eigenstate $|\phi\rangle$ of U , cf-ev allows one to estimate $\langle \psi | U | \psi \rangle$ by preparing the cat state $|\psi_{\text{cat}}\rangle = \frac{1}{\sqrt{2}}(|\psi\rangle + |\phi\rangle)$. We adapt this to estimate $\mathbb{E}_{s \sim p(s)}[\langle s | U_B | s \rangle]$ by sampling different $|s\rangle$ states for each measurement shot, rather than keeping $|s\rangle$ fixed as in the original cf-ev protocol.

To determine an eigenstate of U_B , since Shor and Jordan's method [3] does not prescribe the value of $U_{\sigma_i}|x\rangle$ for non-Fibonacci bitstrings x , we define that $U_{\sigma_i}|000\rangle = e^{i\alpha}|000\rangle$ for some angle α that will be fixed when compiling U_{σ_i} to a circuit. Therefore, we will have that $U_B|0\rangle^{\otimes n} = e^{i\theta_B}|0\rangle^{\otimes n}$ with $\theta_B = w_B\alpha$, and the cat state is given by $\frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + |s\rangle) = V_{\text{cat}}(s)(|+\rangle \otimes |0\rangle^{\otimes n-1})$, where $V_{\text{cat}}(s)$ is the circuit illustrated in Fig. 5(b).

The *full quantum algorithm* runs as follows. First, we sample a computational basis state $|s\rangle$ with probability $p(s)$ [Eq. (4)]. This can be done classically in linear time as shown in Appendix B. Second, we execute the n -qubit circuit shown in Fig. 5(a). Third, we feed the measurement outcome $x_1, x_2, \dots, x_n$ to a function g_1 defined as: $g_1 = 1$ if $x_2 = x_3 = \dots = x_n = 0$, and $g_1 = 0$ otherwise (function g_2 in Fig. 5(a) is discussed later). Finally, we compute the random variable $r = g_1 \cdot z \in \{-1, 0, +1\}$ where $z = (-1)^{x_1}$. As shown in Appendix A, r is a crude estimator of the real or imaginary part of $\mathbb{E}_{s \sim p(s)}[\langle s | U_B | s \rangle]$. By repeating these steps N times and taking the average over samples we obtain a Monte Carlo estimate of the desired

quantity

$$V_{M(B)}(e^{i\frac{2\pi}{5}}) = (-e^{-i\frac{3\pi}{5}})^{3w_B} \phi^{n-2} (R + \epsilon),$$

$$\text{where } R = N^{-1} \sum_{i=1}^N r_i \quad (5)$$

and $\epsilon = \epsilon_{\text{shot}} + \epsilon_{\text{noise}}$.

The real and imaginary parts, $\text{Re}(R)$ and $\text{Im}(R)$, are computed separately by running the above algorithm without ($b = 0$) and with ($b = 1$) the phase gate $S^\dagger$ in the circuit of Fig. 5(a), respectively.

The above protocol can now be minimally tweaked to compute $V_{P(B)}(e^{i(2\pi/5)})$. In this case, instead of sampling $s \sim p(s)$, we always prepare the n -qubit state $|s\rangle = |0101 \dots 10\rangle$ and estimate (see Appendix E):

$$V_{P(B)}(e^{i\frac{2\pi}{5}}) = (-e^{-i\frac{3\pi}{5}})^{3w_B} \phi^{\frac{n}{2}-\frac{3}{2}} \langle s | U_B | s \rangle$$

$$= (-e^{-i\frac{3\pi}{5}})^{3w_B} \phi^{\frac{n}{2}-\frac{3}{2}} (R + \epsilon). \quad (6)$$

The approximation is characterized by two types of errors: the *Monte Carlo shot error* ϵ_{shot} , which induces variance due to a finite number of shots N , and the *gate noise error*, ϵ_{noise} , which induces a bias, which given a quantum computer with certain specifications can be determined.

To implement the circuit shown in Fig. 5 in practice, we optimize the compilation of the 3-qubit matrix $U_{\sigma_i \pm 1}$ of Eq. (1) in terms of the native gate set of the underlying hardware. We specifically aim to minimize the number of 2-qubit gates, as they incur the most errors in most quantum processors. With Quantinuum systems in mind [17], as they currently exhibit the highest fidelities, we choose an ansatz composed of Z-phase gates $R_Z(\theta)$, native 1-qubit gates $U_{1q}(a, b)$ and entangling 2-qubit gates $R_{ZZ}(\chi) = e^{-i(\chi/2)Z \otimes Z}$. We determined the parameters of the ansatz, only requiring the desired action in the Fibonacci subspace, and that $|000\rangle$ is an eigenstate. The resulting circuit is shown in Fig. 4(b), and contains only three 2-qubit gates. We repeated this process for $U_{\sigma_i}^k$ with $2 \leq k < 10$, and replace any consecutive runs of identical braid generators with these optimized circuits. Given another quantum computing platform, this compilation is to be carried out accordingly.

C. Error mitigation

To mitigate the impact of ϵ_{noise} , we can exploit the fact that U_B only acts on a small fraction of the total Hilbert space (that is, the Fibonacci subspace, and the state $|0\rangle^{\otimes n}$). Recall that the algorithm consists of sampling a Fibonacci string $s = s_1, s_2, \dots, s_n$, running the circuit in Fig. 5(a), and then measuring all the qubits to obtain $x = x_1, x_2, \dots, x_n$. We introduce a function g_2 defined as: $g_2 = 1$ if $x_2 = 0$ and $(x_i \oplus s_i) + (x_{i+1} \oplus s_{i+1}) > 0$ for all $i \geq 3$, and $g_2 = 0$ otherwise. From the definition of U_{σ_i} , we have that

$\langle x | U_B | y \rangle = 0$ whenever $x \in \mathcal{F}_n$ and $y \notin \mathcal{F}_n$ or vice-versa. Therefore, if $g_2 = 0$, a hardware error *must* have occurred and that sample can be *discarded* and not included in the estimate R . This is similar to the method of “symmetry constraint” error mitigation discussed in [10]. Under reasonable noise assumptions our technique decreases the bias and increases the variance (see Appendix C), potentially reducing the overall error. We call this technique the *non-Fibonacci error detection*.

Unfortunately, our algorithm is particularly sensitive to coherent phase errors of the form $e^{-i\theta Z/2}$ on the second qubit of U_B . This is because U_σ is diagonal in the first qubit, and is also diagonal in its middle qubit if the first qubit is in the $|0\rangle$ state. Since $|\psi_{\text{cat}}\rangle = \frac{1}{\sqrt{2}}(|s\rangle + |0\rangle^{\otimes n})$ and $s_0 = 0$, then $|\psi_{\text{cat}}\rangle$ has the first qubit always in the $|0\rangle$ state. This implies that U_B is always diagonal on the second qubit, and hence any coherent phase errors on this qubit will commute with U_B and be combined into one large phase error. Since this qubit acts as the control qubit of the Hadamard test (via V_{cat}), a phase error of θ here corresponds to a rotation of the expectation value, so that we estimate $\text{Re}^{i\theta} \approx \mathbb{E}_s[e^{i\theta} \langle s | U_B | s \rangle]$ instead.

The relevant source of such errors on Quantinuum’s trapped-ion platforms is *memory errors*, namely the cumulative effect of errors that happen to qubits while idling. To mitigate this, we can split the algorithm into two phases—first, we estimate $e^{i\theta} R$ for braid B using `cfev`. Then, we estimate $e^{i\theta} R^*$ by running the algorithm for braid B^* by replacing each U_σ with $U_{\sigma^{-1}}$. From this, we can calculate

$$R = \pm \frac{|e^{i\theta} R + e^{i\theta} R^*|}{2} \pm i \frac{|e^{i\theta} R - e^{i\theta} R^*|}{2}, \quad (7)$$

to eliminate the phase error. We call this the *conjugate trick*. Note that the variance of this estimate is half the average variance of $e^{i\theta} R$ and $e^{i\theta} R^*$, so no additional shots are required. Here, R is determined only up to the sign of each component, but this can be disambiguated under the assumption that $|\theta| < \pi/2$. See Appendix C for more details. We have assumed that the coherent error $e^{i\theta}$ is the same in both cases—we believe this assumption is justified since the structure of the conjugate circuits is identical except for the sign flip of the parameters of every gate. In Appendix C, we discuss how the conjugate trick behaves when this assumption is invalid, and we present a method that can still mitigate the phase error under much weaker assumptions about the form of the coherent errors, but cannot determine the signs of the components and may be biased. We call this the *shot-level conjugate trick*.

Finally, a comment regarding choice of protocol and its efficient compilation is in order. The `cfev` protocol has two advantages over the vanilla Hadamard test (see Appendix A for more details on the variants of the Hadamard test). Firstly, by using echo verification,

we have a smaller shot error $\epsilon_{\text{shot}} \propto N^{-\frac{1}{2}}$, as compared to the vanilla Hadamard test where the shot error is $\epsilon_{\text{shot}} \propto (2 - R^2)^{\frac{1}{2}} N^{-\frac{1}{2}}$. Secondly, the vanilla Hadamard test requires implementing many controlled- $U_{\sigma_i^{\pm 1}}$ operations, and using a similar ansatz to Fig. 4(b) these require eight 2-qubit gates each. By comparison, `cfev` uses only U_{σ_i} operations, which require only three 2-qubit gates each, thus reducing the impact of ϵ_{noise} substantially. Overall, by switching to the `cfev` protocol and optimizing the implementation of U_{σ_i} for trapped-ion hardware, we reduce the number of 2-qubit gates required by $\approx 15\times$ and the number of shots required by $\approx 25\%$ as compared to the estimates in Ref. [14]. This, combined with the error mitigation method mentioned above, may allow much larger instances of the problem to be run on Noisy Intermediate-Scale Quantum (NISQ) hardware than was previously anticipated.

Figure 6 shows an example of executing our algorithm on Quantinuum’s H2-2 quantum computer for a braid with 15 strands and 104 crossings that was randomly generated using the method in Sec. III. Applying both the conjugate trick and non-Fibonacci error detection, and sampling 4000 shots per circuit, we observe a relative error of 75% (corresponding to $\epsilon_{\text{noise}} = 0.62$), with a relative standard deviation of 9% (corresponding to $\epsilon_{\text{shot}} = 0.02$). The shot-level conjugate trick improves the relative error to 43% (corresponding to $\epsilon_{\text{noise}} = 0.36$), but at the cost of increasing the relative standard deviation to 14% (corresponding to $\epsilon_{\text{shot}} = 0.06$). In the next section, we will see how this error scales using noisy simulations of a hypothetical NISQ computer.

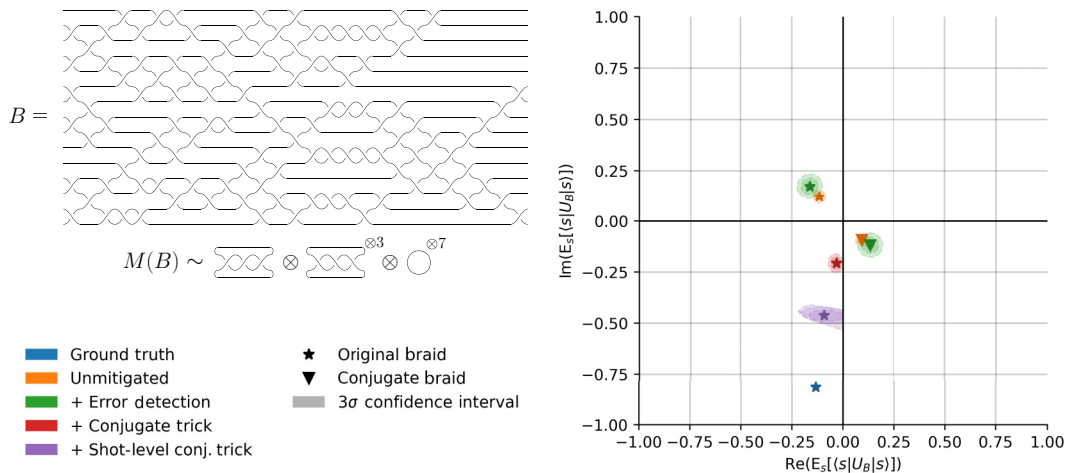


FIG. 6. A demonstration of our algorithm using Quantinuum’s H2-2 quantum computer. We generated a braid B with 104 crossings using the benchmarking procedure described in Sec. III whose Markov closure is topologically equivalent to the tensor product of one left-handed trefoil knot and three right-handed trefoil knots. The resulting quantum circuits acted on 16 qubits with 340 2-qubit gates. Each value was estimated with 4000 shots. Non-Fibonacci error detection improved the estimate by $\sim 25\%$. There is a substantial phase difference between the unmitigated estimate and the ground truth due to coherent memory errors—this can be mitigated by the conjugate trick, but the shot-level conjugate trick described in Appendix C performs better. Our error mitigation and detection techniques are described in Sec. IIC.

D. Less quantum, more advantage

It is believed that $P \subset \text{DQC1} \subset \text{BQP}$, and so DQC1 is “less quantum” than BQP, at least in the worst case. On the other hand, for a given braid, *classically* estimating the trace of the corresponding exponentially large matrix, which solves the DQC1 version of the problem, is generally more challenging than estimating an amplitude of the same matrix, which is required to solve the BQP version. Thus, the experimental results in the next sections focus on the DQC1 formulation of the problem, where our quantum algorithm can achieve “more advantage,” even though our pipeline allows one to search for quantum advantage in the BQP version, as well.

Chen *et al.* [21] define the NISQ complexity class and prove that $\text{BPP}^O \subset \text{NISQ}^O$ and $\text{NISQ}^O \subset \text{BQP}^O$, relative to error-free oracles O . Our hardware assumptions are compatible with the NISQ complexity class, but our DQC1 problem does not rely on an error-free oracle. Further work is needed to understand the relation between NISQ and DQC1. A relevant result by Morimae *et al.* [22] is that the one clean qubit model remains hard to simulate classically for suitably small depolarizing noise applied to the clean qubit.

III. EFFICIENTLY VERIFIABLE BENCHMARK LEVERAGING TOPOLOGICAL INVARIANCE

Benchmarks measure the computing performance, such as error-rate and time-to-solution scaling, on certain tasks. An ideal quantum computing benchmark would be based on a task that requires nontrivial quantum resources,

resembles tasks of practical relevance, and can be efficiently verified by a classical computer while not being efficiently classically computable. An obvious example would be factoring [23], where classical verification simply amounts to multiplying the factors. However, it is difficult to implement such an ideal benchmark on near-term hardware. Benchmarks that rely on random circuits from a universal gate set, often do not resemble circuits used in concrete computational problems. For example, quantum volume [24] is not efficiently classically verifiable and a weakness of mirror benchmarking [25,26] is that classical methods can “cheat” by claiming to have measured the initial state. Application-oriented benchmarks [27,28] are appealing from the practitioner’s point of view, but require the exponentially hard classical simulation of circuits for verification. When the benchmark is efficiently verifiable [29], it often relies on nonuniversal families of quantum circuits, such as Clifford and matchgates, that can be efficiently simulated classically [30–32]. Furthermore, the size of the task used in the benchmark should be under fine control [33]. This is needed if we want to study the *scaling* of performance as quantum circuit depth and width are independently varied. Application-oriented benchmarks based on variational algorithms [34–36] easily decouple circuit width and depth, but, in general, they are affected by an intractable sampling overhead during variational optimization.

Here, we introduce a benchmark based on APPROX-JONES-MARKOV with the appealing feature that it uses a universal gate set relevant to the knot theory problem while being *efficiently verifiable* by classical means. We aim at characterizing the effect of noise on the estimate of $V_{M(B)}(e^{i(2\pi/5)})$, i.e., the scaling of the ϵ_{noise} error with the braid size. Our purpose is to identify the problem sizes that can be solved to a given accuracy and with a given fidelity specification. The links $M(B)$ and $M(B')$ are topologically equivalent iff their braid representatives B and B' can be rewritten into each other via a finite sequence of “Markov” moves (see Appendix D 2, Fig. 14).

We start with a braid B constructed by tensoring k 3-strand braids b_i , so we can classically compute $V_{M(B)}(e^{i(2\pi/5)}) = \phi^{k-1} \prod_{i=1}^k V_{M(b_i)}(e^{i(2\pi/5)})$ at negligible cost. We then augment B by conjugating it with a random braid A , making the braid effectively wider and deeper, respectively, as shown in Fig. 7. This results in a braid $B' = A^{-1}BA$ that by definition shares the same Jones polynomial upon closure, $V_{M(B)} = V_{M(B')}$. Therefore, by measuring the value of $V_{M(B')}$ using our quantum algorithm, and comparing it to $V_{M(B)}$, which is computed efficiently classically, we can measure the error incurred by the quantum computer under test.

If such a B' is constructed naively, then this is very similar to mirror benchmarking [37]. However, in Appendix D 1, we detail a method which generates A and A^{-1} that are inverses to each other, but do not have similar

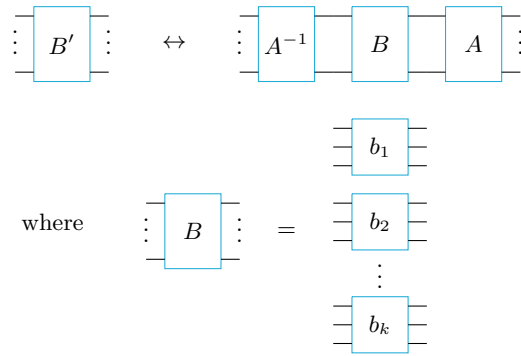


FIG. 7. Augmenting a $3k$ -strand braid $B = \otimes_{i=1}^k b_i$ by conjugation with a random braid A , and obtaining braid B' . Braid B is composed of random 3-strand braids $b_1, \dots, b_k$, so its Jones polynomial can be evaluated classically exactly in negligible time as $V_{M(B)}(e^{i(2\pi/5)}) = \phi^{k-1} \prod_{i=1}^k V_{M(b_i)}(e^{i(2\pi/5)})$. See Appendix D 1 for details. B is generated so that the magnitude of $V_{M(B)}$ is approximately uniformly distributed. The random braid A and its inverse A^{-1} are generated to be inverse to each other but not mirror images.

structures. In particular, A is generated randomly and is much longer than A^{-1} . It appears close to a random braid in structure, except that the signs of the crossings are picked such that A^{-1} can be constructed with a shallow braid. This results in overall braids B' that appear similar to random brick-wall braids.

Furthermore, since we wish to characterize ϵ_{noise} rather than ϵ_{shot} , which has a known distribution, we pick the braids b_i from a probability distribution that is set up so that the distribution of $|\mathbb{E}_s[\langle s|U_B|s \rangle]|$ is as uniform as possible over the range $[0, 1]$. If the b_i were picked uniformly randomly, then the distribution of $|\mathbb{E}_s[\langle s|U_B|s \rangle]|$ would concentrate exponentially around 0 as the number of strands increases (since the average value of $|\mathbb{E}_{s_i}[\langle s_i|U_{b_i}|s_i \rangle]|$ is less than one), which would require an exponentially increasing number of shots to ensure that ϵ_{shot} is not the dominant error.

Therefore, via augmentation, we can obtain benchmarking sets of braids with varying numbers of strands, crossings, and depths. To characterize the scaling of the error with increasing number of crossings for potential NISQ computers, we constructed a dataset of 24k braids of between 10 and 15 strands and 50 to 600 crossings. We classically simulated `cfev` with non-Fibonacci error detection on these braids and calculated the relative error in the output. We used a large number (10k) of shots per braid, so that this error characterizes ϵ_{noise} accurately.

The simulation modeled the native gate set used in the ansatz in Fig. 4(b) with errors inserted as depolarizing channels on 1- and 2-qubit gates with probabilities ϵ_{1q} and ϵ_{2q} . We tested both $\epsilon_{2q} = 5 \times 10^{-4}$ and $\epsilon_{2q} = 1 \times 10^{-4}$ as the dominant error, and set $\epsilon_{1q} \approx \epsilon_{2q}/10$. SPAM bit-flip errors were also included with $\epsilon_{\text{SPAM}} \approx \epsilon_{2q}$ per qubit. We

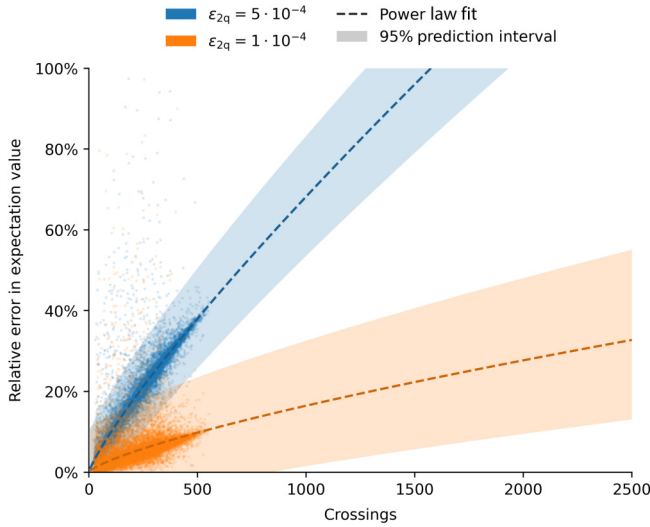


FIG. 8. For a benchmark set of 24k randomly generated braids and two different rates of depolarizing 2-qubit errors, this shows how the relative error $|R - \mathbb{E}_s[\langle s | U_B | s \rangle]| / |\mathbb{E}_s[\langle s | U_B | s \rangle]|$ between the estimated value (over 10k shots) and the exactly known value depends on the number of crossings (and hence 2-qubit gates).

did not attempt to model coherent effects such as memory errors since this varies widely by platform and technology. See Appendix F for more details.

In Fig. 8, we show how the relative error for this benchmark set scaled with the number of crossings, along with a power-law extrapolation to larger numbers of crossings. We observed only weak scaling in terms of number of strands and depth when fixing the number of crossings (Fig. 15). This aligns with the expectation that errors from a depolarizing noise model ought to depend mainly on the number of gates. These results inform us about the maximum crossing number, or equivalently gate count (specifically 2-qubit gates since they are the dominant noise factor [17]), we can reach while remaining within a desired error budget.

These simulations should be broadly representative of NISQ devices with similar error rates, at least to the extent that the depolarizing channel is accurate, or to the extent that the circuit structure sufficiently mixes any structure in the actual error channel. This picture can be ensured by employing randomized compiling techniques, and we note that this should be approximately realized in our experiments on random braids [38]. There is one notable exception, which is coherent phase effects, for example, due to memory errors (which are especially apparent on quantum computers with a relatively slower gate speed, such as trapped-ion devices) or cross-talk between qubits (which are more common on superconducting devices).

For the results of these simulations to be applicable, we assume that the coherent portion of this error is negligible,

either by applying the conjugate trick or by mitigating it at a hardware level, and that any remaining coherent errors can be incorporated into an effective 1- and 2-qubit depolarizing error rates. For example, for trapped-ion technology, this can be achieved via dynamical decoupling [39]. This is a reasonable assumption as we expect rates of coherent errors to be improved at a hardware level before the 2-qubit gate-error rates considered here are achieved.

Since these results are obtained via simulation, we must *extrapolate* to larger sizes. If one uses an actual quantum backend which is challenging to simulate classically, one would perform an *interpolation* on similarly obtained data. Further, note that this benchmark will take significantly longer to run on an actual device as clock speeds of quantum processors are orders of magnitude lower than those of classical processors, but one may compensate with fewer braids in the benchmark set or fewer shots, in which case it may be beneficial to apply a technique such as deconvolution [40] to separate the effects of ϵ_{shot} and ϵ_{noise} . While we have performed a relatively naive extrapolation by gathering as much data as is feasible given the constraints of classical simulation, our benchmarking pipeline can be used as is with other extrapolation methods, or alternative data (for instance, combining large-scale classical simulation with a few data points from quantum hardware).

IV. RESOURCE ESTIMATES FOR NEAR-TERM QUANTUM ADVANTAGE

We now demonstrate how one may use the algorithmic tools introduced above to perform resource estimates for APPROX-JONES-MARKOV, for our quantum algorithm and for classical algorithms that either simulate a quantum circuit or approach the problem in a nonquantum-related way. Regarding our quantum algorithm, given a braid of certain size, i.e., number of strands $n - 1$ and crossing number c , with a required additive error ϵ , we can infer the required number of shots $N = \epsilon_{\text{shot}}^{-2}$ from $\epsilon_{\text{shot}} = \epsilon - \epsilon_{\text{noise}}$. Then, N determines the time to solution of the quantum algorithm (see Table I), noting that ϵ_{noise} depends on the braid size.

From this, we estimate the braid *size* for which we might obtain quantum advantage, by analyzing the runtime of the most competitive classical algorithm returning the solution to the same error. We note that this method has limitations, and we must make some assumptions—these are explained in Sec. IV D. Only if these assumptions are satisfied can we draw concrete conclusions about the existence of a quantum advantage for generic braids. If this is not the case, we can still use the method of Sec. III as a flexible benchmark of noisy quantum hardware.

A. Classical simulations of the quantum algorithm

We can construct a tensor network to exactly evaluate $V_{M(B)}(e^{i(2\pi/5)})$ by representing the weighted trace

TABLE I. Approximate scaling in runtime and space of the algorithms we consider as a function of n qubits or equivalently strands, c crossings, N shots, maximum bond dimension χ , and ϵ_{2q} 2-qubit gate error. The parameters $\alpha \approx 1.04$, $\beta \approx 1.05$, $\gamma \approx 1.01$, and $\mu, \nu \approx 1.03$ are representative numbers taken from the extrapolations shown in Figs. 10(a) and 11(b). Note that since $c \sim 10^3$, the difference between γ^c and β^c is very large (e.g., 10^{16}).

Algorithm	Space	Time	ϵ_{shot}	$1 - \epsilon_{\text{noise}}$
cfev	n	$(c + n)N$	$N^{-1/2}$	$(1 - \epsilon_{2q})^{3c}$
tn-proj	α^c	β^c	...	...
mpo-proj	$n\chi^2$	$cn\chi^3$	...	$n\chi^2\gamma^{-c}$
sv-shor	ϕ^n	$c\phi^n N$	N^{-1}	...
bracket	μ^c	ν^c	...	...

of Ref. [3] as the usual trace $\text{tr}(U_B)$ and insert the weights using 2-qubit projectors onto the Fibonacci subspace, $P_{\mathcal{F}}^{\partial_0} = |01\rangle\langle 01|$, $P_{\mathcal{F}} = |01\rangle\langle 01| + |10\rangle\langle 10| + |11\rangle\langle 11|$, $P_{\mathcal{F}}^{\partial_1} = |10\rangle\langle 10| + \phi(|01\rangle\langle 01| + |11\rangle\langle 11|)$, as shown in Fig. 9 for a five qubit example. From this, we construct three classical algorithms. Firstly, the tensor network can be contracted directly and exactly, we call this method `tn-proj`. Since it cannot exploit the sparsity of U_B (which is only defined on a small subspace of the total Hilbert space), but only its structure, we would expect this to be very inefficient in general.

Alternatively, we consider the approximating $\text{tr}(U_B P_{\mathcal{F}})$ using the Hutch++ stochastic trace estimator [42], which estimates such traces to relative error ϵ as a linear combination of $O(\epsilon^{-1})$ amplitudes $\langle \psi | U_B P_{\mathcal{F}} | \psi \rangle$. Each amplitude is computed using a statevector simulation, which can

effectively exploit the sparsity of U_B by storing only the coefficients corresponding to bitstrings in $\mathcal{F}$, thus lowering the complexity of computation from $O(2^n)$ to $O(\phi^n)$. This is effectively a statevector simulation of Shor and Jordan's algorithm, so we call it `sv-shor`.

Finally, we consider approximating the contraction of `tn-proj` by constructing a matrix product operator (MPO) from $U_B P_{\mathcal{F}}$, from which the trace can be computed in polynomial time with respect to the bond dimension χ of the MPO by exact contraction. We call this method `mpo-proj`. By fixing a maximum bond dimension and compressing bonds using the usual singular-value method, we can control the approximation cost.

The scaling of the χ required to meet the desired error budget can be characterized empirically using the same benchmark method used to characterize the quantum algorithm (Sec. III). As in `sv-shor`, this method can take advantage of the sparsity of U_B by discarding singular values which are exactly zero. In practice, we find that the maximum bond dimensions are often Fibonacci numbers, indicating that this is indeed occurring (see Fig. 19).

B. Classical algorithms for the Jones polynomial

Various classical methods to compute the Jones polynomial have been proposed, however, there are comparatively few results on the complexity of these algorithms at scale, either empirically or theoretically. As discussed further in Appendix F 3, we believe the most competitive method is due to Kauffman [1, p. 125] which computes the Jones polynomial, via the bracket polynomial, by contracting a tensor network defined over a polynomial ring (this network is distinct from the one considered by `tn-proj`). We call this method `bracket`.

C. Empirical characterization

To determine the exact scaling of `tn-proj`, `sv-shor`, `mpo-proj`, and `bracket`, we benchmark them empirically on a dataset of braids generated according to the method in Sec. III. We aim to compare the time required for each classical method to approximate $V_{M(B)}(e^{i(2\pi/5)})$ to the same relative error as `cfev`.

We assume two different classical hardware targets: an A100 80 GB GPU [45] to represent typical maximum performance of a single compute node, and Frontier [46] representing typical maximum performance available in a large high-performance supercomputer. In both cases, we assume perfect parallelism and utilization, and calculate the time required as the number of 32-bit floating point operations (FLOPS) divided by theoretical maximum FLOPS per second. For the quantum algorithms, we assume the time to solution scales with the depth of the quantum circuit as 30 ms per layer (this is similar to existing trapped-ion systems [47, Table I]), and that the relative error scales as in Fig. 8.

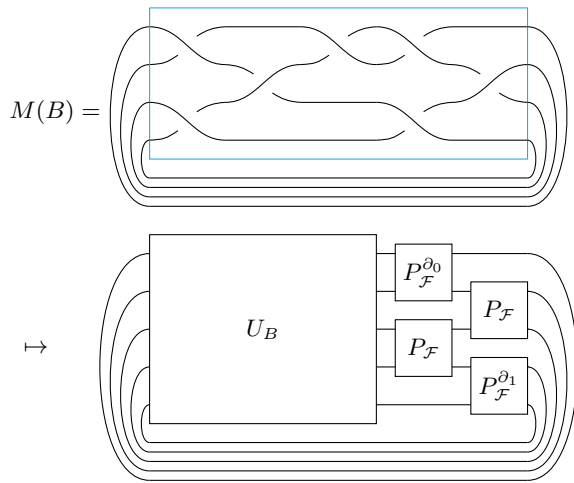


FIG. 9. The Markov closure $M(B)$ of a braid B , and the corresponding tensor network `tn-proj` that computes the weighted trace $\sum_s P(s) \langle s | U_B | s \rangle$, with the 2-qubit projectors $P_{\mathcal{F}}$ inserting the appropriate weights.

The dataset contained 10k braids of up to 600 crossings and between 10 and 30 strands. When constructing large instances, it is not sufficient to scale up the number of crossings or strands individually while keeping the other fixed, since this will lead to braids that are too shallow or too deep. Therefore, in order to extrapolate to larger sizes than we could benchmark in practice, we computed the principal component analysis of the dataset in terms of crossings, depth, and strands, and extrapolated along the principal component, so that crossings and strands increase together.

For `tn-proj` and `bracket`, we used Cotengra [48] to calculate optimized contraction paths and estimate the contraction cost. For `sv-shor`, we utilize the estimates of distributed statevector computations from [49] to calculate the cost per crossing. We use an optimistic QBF factor of 0.25 (a parameter related to the FLOPS per statevector entry per gate) so as to handicap `cfev` as much as possible. The number of amplitudes N to calculate was taken as $\lceil 4\epsilon_{\text{noise}}^{-1} \rceil$ as in [42, Theorem 10].

Finally, for `mpo-proj`, we used Quimb [50] to implement MPO evolution. For each braid, we first ran the

algorithm without compressing any nonzero singular values, so that we could observe the runtime scaling for an error-free computation. These results are shown in Fig. 10(b). Then, we ran the same computation with the bond dimension limited to $\chi \leq 2^k$ for $k \in [3, 10]$, to estimate the error resulting from a given level of compression. In Fig. 10(a), using the known value of the Jones polynomial for each braid, we computed the relative error as it depends on the maximum available memory. Taking the 97.5% lowest quantile of this relationship (that is, the lower bound of a 95% confidence interval) to handicap `cfev` as much as possible, we computed the maximum value of χ needed to achieve the same relative error as `cfev`, and using the theoretical FLOP counts of each operation [51, Table 3.13] we computed the time to solution across the extrapolated dataset for both $\epsilon_{2q} = 1 \times 10^{-4}$ and $\epsilon_{2q} = 5 \times 10^{-4}$.

The results are shown in Fig. 11(a). We can see that for $c \geq 1100$, the relative error of the quantum computer for $\epsilon_{2q} = 5 \times 10^{-4}$ is so high so that the bond dimension required to approximate it (according to Fig. 10) drops to zero. This happens while the time to solution

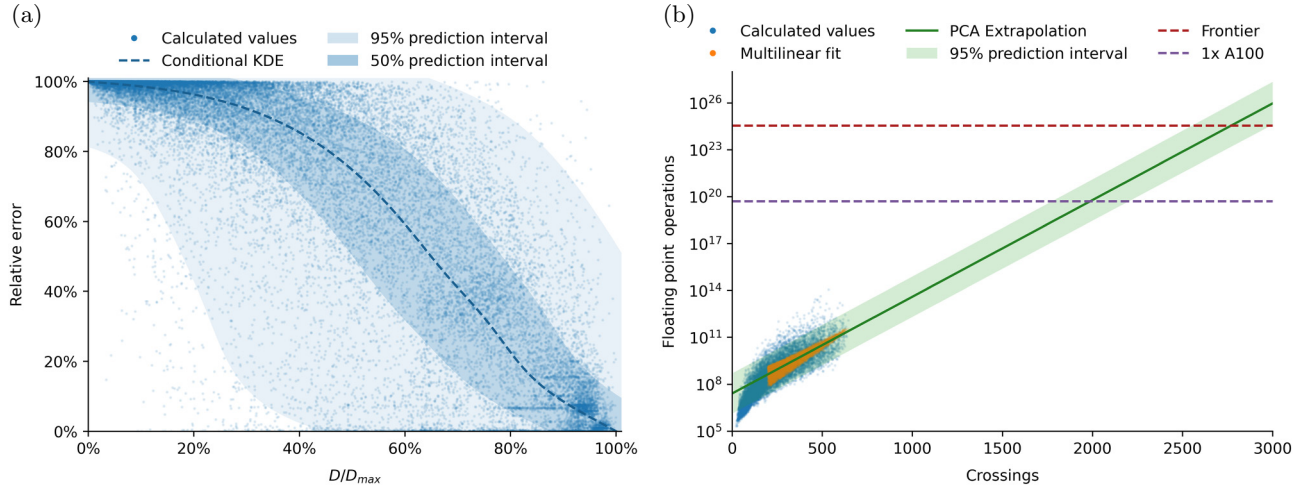


FIG. 10. Studying the error introduced by compression of the matrix product operator computation through limiting the bond dimension, as well as the scaling of runtime with number of crossings when the bond dimension is large enough that the computation is error-free. The calculated values are from a dataset constructed using the benchmark method in Sec. III and comprising 10k braids of up to 600 crossings and 30 strands. (a) A comparison of the relative error in $V_{M(B)}(e^{i(2\pi/5)})$ achieved by the `mpo-proj` algorithm as it depends on the maximum required memory D , normalized to the case D_{max} where χ is set very large (so that the relative error is zero). We can see that as the available memory decreases, the relative error increases. Note that above a threshold of $\sim 80\%$ relative error, $D = 0$ is sufficient with at least 5% confidence, so we consider the cost of executing `mpo-proj` as effectively zero. The prediction intervals and regression line were obtained via Nadaraya-Watson conditional kernel density estimation [41]. See Fig. 16 for how the amount of computation required depends on the required memory and number of crossings. (b) The number of floating-point operations required to evaluate $V_{M(B)}(e^{i(2\pi/5)})$ using the `mpo-proj` method for the same benchmark set. The bond dimension $\chi = \chi_{\text{max}}$ was allowed to grow as large as needed to avoid any lossy compression. The extrapolation is a multilinear fit of $\log_{10}(\text{FLOPS})$ against the number of strands $n - 1$, crossings c , and depth of the braid for $c \geq 200$. It is extrapolated along the principal component of the dataset to represent possible performance for larger datasets generated in the same way. The lines labeled “Frontier” and “1x A100” represent one month of computation on the respective hardware, for scale. See Fig. 18 for the equivalent plot in terms of memory consumption.

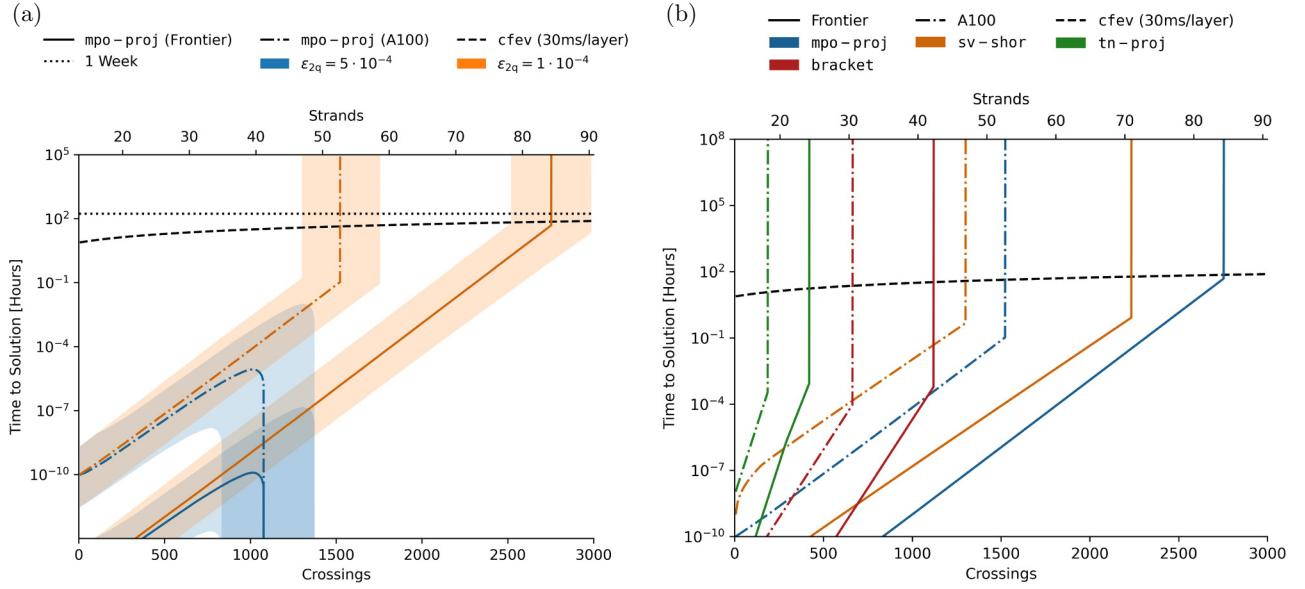


FIG. 11. Time comparisons for calculating $V_{M(B)}(e^{i(2\pi/5)})$ on the same benchmark set as in Fig. 10, to the same relative error achieved by cf-ev, as it depends on the number of crossings c and the 2-qubit error rate ϵ_{2q} . The solid lines represent computation on the Frontier cluster, while the dot-dashed lines represent the same computation on a single A100 GPU. The lines vertically downward represent the points where the relative error of cf-ev exceeds the 80% threshold given by Fig. 10, whereas the lines vertically upward represent the points where required memory exceeds the available memory of the classical hardware. (a) The time needed for the mpo-proj method, as it depends on the number of crossings c and the 2-qubit error rate ϵ_{2q} . We can see that for $\epsilon_{2q} = 5 \times 10^{-4}$, the relative error of cf-ev grows large enough that mpo-proj is always faster than cf-ev. Conversely, for $\epsilon_{2q} = 1 \times 10^{-4}$, the memory limit is reached before this point. Even if this were not the case, we would expect cf-ev to be faster than mpo-proj on Frontier when $c \geq 2800$. (b) A comparison of the time needed for each classical method when $\epsilon_{2q} = 1 \times 10^{-4}$, as it depends on the number of crossings c . We can see that mpo-proj is the most competitive classical method, although the memory limit of Frontier is reached before it becomes slower than cf-ev. See Figs. 17–21 for plots of the floating point operations, memory consumption, and time to solution of each classical method individually.

for mpo-proj is still much smaller than that of cf-ev, and hence it is likely not possible to achieve quantum advantage with $\epsilon_{2q} = 5 \times 10^{-4}$.

In Fig. 11(b), we compare the time-to-solution scaling for all methods when $\epsilon_{2q} = 1 \times 10^{-4}$. We observe that mpo-proj is the most competitive classical method, but that all methods hit the memory limits of the corresponding hardware target before they become slower than the quantum computer. Above these limits, the computation would likely not be possible on that classical hardware. Even if this were not the case, our results suggest that mpo-proj executed on Frontier will likely be slower than cf-ev when $c \geq 2800$, at which point the relative error of cf-ev would be $\sim 40\%$.

Furthermore, at large scales, cf-ev may be more energy efficient. For example, [44] pessimistically estimates the energy consumption of Quantinuum’s H2 quantum computer at 75 kW. By comparison, the most efficient classical computing cluster as of November 2024 is the JEDI system at the Jülich Supercomputer Center [43], with an efficiency of 2.6×10^{17} FLOPS/kWh. Assuming these values, we expect that cf-ev becomes more energy-efficient than mpo-proj around $c \geq 2400$, as shown in Fig. 12.

D. Limitations

In the previous section, we determined the size of braid for which the cf-ev method may be faster or more energy-efficient than the competing classical methods, when evaluated on the set of benchmark braids. A priori, there is no reason to conclude that the advantage extends to braids outside of this set. Indeed, the benchmark braid instances are not generic or typical, and they have a special structure by virtue of their construction.

However, it is reasonable to assume that, for generic classical algorithms, generic braids without the same special structure will be at least as hard to evaluate. Under this assumption, the results of the previous section would provide a lower bound on the classical hardness of a generic family of braids. In Appendix F4, we show that the mpo-proj method may be able to exploit some of the structure of the benchmark braids. This is encouraging, as it suggests that generic braids may be even harder to solve with the classical methods tested here. In contrast, the runtime of the cf-ev quantum algorithm does not depend on the family of braids, but only on their size. Thus, given the aforementioned lower bound on the classical hardness of generic braids, the estimated crossover point where we

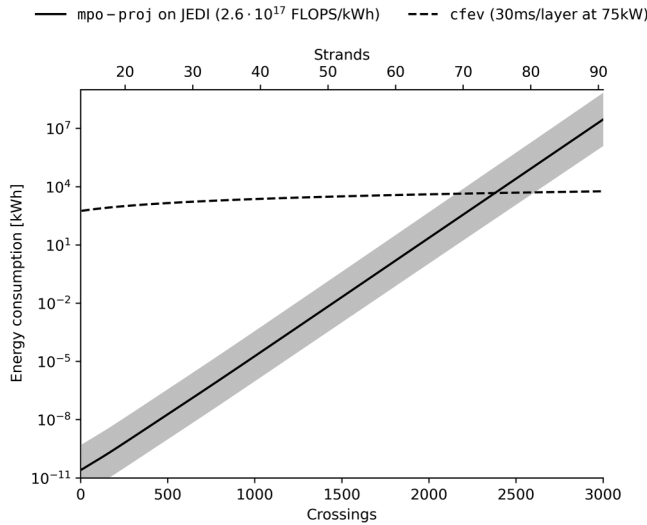


FIG. 12. A comparison of the energy consumption between the most competitive classical algorithm `mpo-proj`, assuming it is running on Jülich’s JEDI supercomputer [43] (without considering memory limitations) with an efficiency of 2.6×10^{17} floating point operations/kWh, and our quantum algorithm `cfev`, assuming the same estimated 75 kW power usage as Quantinuum’s H2 quantum computer [44].

expect a quantum advantage for the benchmark braids is an estimate of the upper bound for where we might see a quantum advantage for generic braids.

It is also important to stress that our conclusions here are based on extrapolations from necessarily limited data (up to $c = 600$). The main obstacle to benchmarking the classical methods (especially `mpo-proj`) on larger braids was the amount of memory available on a single GPU and the computation time required. This is promising, as it indicates these problems are genuinely hard to solve classically, but conversely it means that if one wanted to be certain that a particular instance of this problem is faster on a quantum computer than the classical equivalent, then more compute and larger-scale datasets would be needed.

On the one hand, generating random braids is reminiscent of generating random circuits to demonstrate quantum computational supremacy. On the other hand, the problem of approximating the Jones polynomial is not only a complete problem for quantum complexity classes but also potentially interesting in knot theory, a rich field of research in low-dimensional topology. See Appendix F for further details on the estimates, extrapolation, and benchmark results.

V. END-TO-END PIPELINE

We have introduced a comprehensive, configurable pipeline enabling the identification of problem sizes exhibiting quantum advantage, which serves as the main contribution of this work, beyond the introduction of the

most competitive classical and quantum algorithms for the Jones polynomial. Our benchmark can characterize the error scaling for both a quantum algorithm running on a noisy quantum computer and classical approximation algorithms running on state-of-the-art classical hardware.

Code for the end-to-end implementation of our quantum algorithm, the efficiently verifiable benchmark, as well as the competitive classical algorithms, can be found in an online repository [52]. Given an input braid or link, this implementation can produce quantum circuits to be executed and perform the necessary classical preprocessing and postprocessing. Additional tools for braid simplification and to perform the compilation of the braid generators for new architectures are also included.

Given the most efficient quantum and classical algorithms for solving a problem, the search for a problem instance for which it is advantageous to use a quantum processor over a classical computer is multidimensional. These dimensions are the error on the estimated quantity, the wall-clock time taken to carry out the computation, the specifications of the quantum computer (2-qubit gate fidelity and depth-1 circuit time), and the analogous specifications of the classical computer (FLOPS per second). Note that achieving a quantum gate fidelity above a certain value may require error detection, error mitigation, or even error correction, which can significantly impact the clock speed of the quantum processor. The specifications of the quantum computer may vary widely over different platforms, but our approach is agnostic and applicable to all. To detect quantum advantage, one then fixes a desired error budget and searches for an instance for which time to solution on a quantum computer with given specifications is lower than on a state-of-the-art classical computer. Alternatively, given an instance of interest, one may search for the quantum system requirements for achieving a time-to-solution advantage. We have also demonstrated how energy efficiency advantage can be quantified and estimated. The reconfigurability of our pipeline enables the exploration of this multidimensional space. More generally, with our rigorously quantitative and practical approach, we aim to shift the quantum advantage discussion toward *problem instances* rather than problem classes.

Since we have reduced the search for advantage instances, given hardware specifications, to a complex optimization problem, we envision that the search for such quantum-easiest classically-hardest instances can readily be approached with state-of-the-art AI methods for scientific discovery [53,54], which we leave for future work. For instance, given a particular braid, finding a smaller but topologically equivalent braid that may be easier to solve classically is a difficult task that has received some attention [55], including by reinforcement learning [56].

Finally, we stress that even though we focused on the APPROX-JONES-MARKOV for benchmarking and

resource estimation, the problem APPROX-JONES-PLAT can also be approached similarly. The benchmark set of braids can be converted to Plat-closed braids [see Fig. 1(b)], and by construction, their Jones polynomials are invariant under this conversion. Similarly, `mpo-proj` can be adapted to use matrix product states rather than MPOs, and we would expect it to remain the most competitive classical baseline.

VI. DISCUSSION

We have provided rigorous resource estimates for the DQC1-complete problem of approximating the Jones polynomial of a Markov-closed braid, up to an exponentially large-scale factor, at the fifth root of unity when solved on a noisy BQP machine, i.e., a digital quantum computer. We optimally compiled the control-free version of the Hadamard test to reduce the 2-qubit gate count, and used echo verification to reduce the number of shots. Our extrapolated results suggest that a quantum processor equipped with ≈ 100 qubits, $\epsilon_{2q} \leq 10^{-4}$, and depth-1 circuit-times on the order of tens of milliseconds is promising for *demonstrating quantum advantage in terms of time to solution*. The classical algorithms used as baselines include classical simulations of the quantum circuits, which are in general more challenging than they are for the BQP-complete formulation of the problem which regards Plat-closed braids. Therefore, we have used a “less quantum” problem to gain “more advantage.”

Our problem-specific efficiently verifiable benchmark can be used to characterize the error scaling of a noisy digital quantum computer, as well as the heuristic performance of approximate classical methods. Thus, we provide practical tools for locating the advantage boundary given the current state-of-the-art classical methods. Once a quantum computer with sufficient specifications is available, as the one assumed in this work, a quantum advantage for this problem may be identified.

We also showed that the BQP-complete version of the problem, where the braid is instead Plat-closed, can be solved using the same quantum protocol with a simple modification. Therefore, our end-to-end pipeline can be used to identify the minimum requirements for quantum advantage for evaluating the Jones polynomial for Plat-closed braids as well.

Future work involves rigorous resource estimates in the fault-tolerant regime, compiling the algorithm to the appropriate gate set and assuming the relevant overheads and realistic methods. This analysis will inevitably be required for obtaining better precision, which depends on the fidelity of the quantum protocol. The current belief is that gate errors below 10^{-4} will require more error mitigation and perhaps error correction. In the near term, memory errors may be addressed via decoherence-free subspace codes, and given that our quantum algorithm operates only

in a small subspace of the full Hilbert space, it may be possible to exploit this to decrease the cost of error correction, for example by allowing smaller code distances.

Our algorithm and resource estimation pipeline can also be generalized to other nonlattice roots of unity $e^{i2\pi/k}$, for $k \in \mathbb{N}$, where other unitary representations of the braid group would need to be compiled (see Appendix G for a brief sketch) [57]. With the ability to approximate the value of the Jones polynomial at roots of unity, one could distinguish knots and links with high probability, a computational tool that practitioners in the field of knot theory could utilize. Consequently, our algorithm could be adapted to compute the colored Jones polynomial [1]. Our general approach for benchmarking and diagnosing system requirements for quantum advantage in practice also applies to other more powerful invariants such as the Khovanov homology [58].

ACKNOWLEDGMENTS

We thank David Amaro and Etienne Granet for comments on the manuscript, and Adam Connolly, Natalie Brown, and Harry Buhrman for discussions. We thank Julia Cline, Joan M. Dreiling, Alex Hall, Aaron Hankin, Azure Hansen, Nathan Hewitt, Chris Langer, Elliot Lehman, Dominic Lucchetti, and the entire operations team for technical support during the implementation on the Quantinuum H2-2 quantum computer. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC Award NERSC DDR-ERCAP0029825.

DATA AVAILABILITY

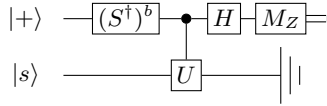
The data that support the findings of this article are openly available [52].

APPENDIX A: THE CONTROL-FREE HADAMARD TEST WITH ECHO VERIFICATION

In this work, we employ Hadamard tests as key algorithmic primitives. Such protocols are designed to estimate $\langle s | U | s \rangle$ for a given unitary U and initial state $|s\rangle$. The execution of Hadamard test circuits on NISQ computers is affected by two main sources of errors: shot noise ϵ_{shot} , which contributes to the variance of any estimator, and gate noise ϵ_{noise} , which contributes to the bias of any estimator. It is therefore important to optimize the estimator and reduce the number of qubits and gates in the circuit. We now briefly summarize `ht`, and its variant `cfev` that forms the basis for our algorithm. The reader may refer to Ref. [8] for a fully detailed exposition.

`ht` encodes the relevant information in the relative phase between the $|0\rangle$ and $|1\rangle$ states of the ancilla qubit

by means of the following circuit:



The relevant information is then extracted by measuring the ancilla and averaging the binary outcomes. By setting $b = 0$ one can estimate $\text{Re}(\langle s | U | s \rangle)$, and setting $b = 1$ allows one to estimate $\text{Im}(\langle s | U | s \rangle)$.

Now, suppose that an eigenvalue/eigenstate pair $U|\lambda\rangle = e^{i\theta}|\lambda\rangle$ is known, and that the initial state is orthogonal to the eigenstate, $\langle s | \lambda \rangle = 0$. Then one can *remove the ancilla* qubit in *ht* and encode the relevant information in the relative phase between $|\lambda\rangle$ and $|s\rangle$. This is achieved using a cat-state preparation circuit, $V_{\text{cat}}(s)(|+\rangle \otimes |0\rangle^{\otimes n-1}) = \frac{1}{\sqrt{2}}(|\lambda\rangle + |s\rangle)$. A single qubit rotation gate $R_Z(\theta)$ is used to compensate the phase of the known eigenvalue. Importantly, the desired unitary, U , can be compiled using fewer entangling gates than its controlled version, and so the gate-error ϵ_{noise} is smaller. Further, an error mitigation trick called *echo verification* reduces the shot noise ϵ_{shot} . Since the state preparation circuit for the initial state, $V_{\text{cat}}(s)$, is known, at the end of the circuit its inverse, $V_{\text{cat}}^\dagger(s)$, is applied. The measurement outcomes are postprocessed by

a Boolean function g which checks whether the initial state is recovered or not. This information is then combined with measurements on the ancilla to form an unbiased estimator with reduced variance. This leads to the control-free Hadamard test with echo-verification (*cf ev*) protocol that we use in this work.

Let us now specialize the discussion to our specific use case. Recall that for the estimation of both Markov and Plat closures of braid B , we have constructed a unitary U_B with known eigenvector $|0\rangle^{\otimes n}$ and corresponding eigenvalue $e^{i\theta_B}$. Moreover, the initial state $|s\rangle$ always encodes a bitstring s , so there exist an efficient circuit $V_{\text{cat}}(s)$ for cat-state preparation. This leads us to *cf ev* circuit shown in Fig. 5(a). As we show next, this protocol has better scaling in both ϵ_{shot} and ϵ_{noise} errors compared to *ht*. Moreover, the final measurement reveals whether the main register is in the Fibonacci subspace. This is used for our error detection strategy explained in Appendix C.

For simplicity of notation we drop subscripts B and cat and denote the initial state as $|s\rangle = |01s_3s_4\dots\rangle$. Without loss of generality we assume that $\theta_B = 0$, i.e., that the known eigenvalue is 1. The action of the cat-state preparation circuit V is such that $V|0x_2x_3\dots\rangle = |x_20x_3\dots\rangle$ and $V|1x_2x_3\dots\rangle = |x_21(x_3\dots \oplus s_3\dots)\rangle$. Note that $V|0\rangle^{\otimes n} = |0\rangle^{\otimes n}$ and $V|10\dots\rangle = |s\rangle$. Therefore,

$$|\psi\rangle = UV(H \otimes I^{\otimes n-1})|0\rangle^{\otimes n} = \frac{1}{\sqrt{2}}UV(|00\dots\rangle + |10\dots\rangle) = \frac{1}{\sqrt{2}}U(|0\rangle^{\otimes n} + |s\rangle) = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + U|s\rangle).$$

At the end of the *cf ev* circuit, the computational basis has the following amplitudes:

$$\begin{aligned} \langle\psi| V^\dagger(H \otimes I^{\otimes n-1}) |x_1x_2x_3\dots\rangle &= \frac{1}{\sqrt{2}} \langle\psi| V^\dagger(|0x_2x_3\dots\rangle + (-1)^{x_1} |1x_2x_3\dots\rangle) \\ &= \frac{1}{\sqrt{2}} \langle\psi| (|x_20x_3\dots\rangle + (-1)^{x_1} |x_21(x_3\dots \oplus s_3\dots)\rangle) \\ &= \frac{1}{2} ((00\dots||x_20x_3\dots) + \overbrace{(-1)^{x_1} \langle 00\dots|x_21(x_3\dots \oplus s_3\dots)\rangle}^{=0}) \\ &\quad + \overbrace{\langle s| U^\dagger |x_20x_3\dots\rangle}^{=0} + (-1)^{x_1} \langle s| U^\dagger |x_21(x_3\dots \oplus s_3\dots)\rangle). \end{aligned}$$

Note that U maps Fibonacci strings starting with 01 to Fibonacci strings starting with 01. Thus in the last step the third term is zero because $U|s\rangle$ is a Fibonacci string starting with 01, while $|x_20x_3\dots\rangle$ is not. Using this property of U we can also write

$$\langle\psi| V^\dagger(H \otimes I^{\otimes n-1}) |x_1x_2x_3\dots\rangle = \begin{cases} \frac{1}{2}(1 + (-1)^{x_1} \langle s| U^\dagger |s\rangle) & \text{if } x_2 = 0 \text{ and } x_3\dots = 0 \quad (1), \\ \frac{1}{2}(-1)^{x_1} \langle s| U^\dagger |x_21(x_3\dots \oplus s_3\dots)\rangle & \text{if } x_2 = 0 \text{ and } x_3\dots \oplus s_3\dots \text{ is Fibonacci} \quad (2), \\ 0 & \text{otherwise} \quad (3). \end{cases} \quad (\text{A1})$$

Let us now define two postprocessing functions g_1 and g_2 for the measurement outcomes. If the measurement detects case (1), we set $g_1 = 1$, $g_2 = 1$. If the measurement detects case (2), we set $g_1 = 0$, $g_2 = 1$. Finally if the measurement detects case (3), we set $g_1 = 0$, $g_2 = 0$. The amplitude of case (3) is zero, meaning that this event should never happen (we will utilize this information for error mitigation in Appendix C). Let us define the ternary random variable $r = (-1)^{x_1} \cdot g_1 \in \{-1, 0, 1\}$. From the Born rule, the probabilities of the three outcomes are

$$\begin{aligned} P(r = +1) &= P(x_1 = 0, x_2 = 0, x_{3..} = 0) = \frac{1}{4} |1 + \langle s | U^\dagger | s \rangle|^2 = \frac{1}{4} + \frac{|\langle s | U | s \rangle|^2}{4} + \frac{\text{Re}(\langle s | U | s \rangle)}{2}, \\ P(r = -1) &= P(x_1 = 1, x_2 = 0, x_{3..} = 0) = \frac{1}{4} |1 - \langle s | U^\dagger | s \rangle|^2 = \frac{1}{4} + \frac{|\langle s | U | s \rangle|^2}{4} - \frac{\text{Re}(\langle s | U | s \rangle)}{2}, \\ P(r = 0) &= 1 - P(r = +1) - P(r = -1) = \frac{1}{2} - \frac{|\langle s | U | s \rangle|^2}{2}. \end{aligned}$$

The first two raw moments of r are

$$\begin{aligned} \mathbb{E}[r] &= \left[\frac{1}{4} + \frac{|\langle s | U | s \rangle|^2}{4} + \frac{\text{Re}(\langle s | U | s \rangle)}{2} \right] - \left[\frac{1}{4} + \frac{|\langle s | U | s \rangle|^2}{4} - \frac{\text{Re}(\langle s | U | s \rangle)}{2} \right] = \text{Re}(\langle s | U | s \rangle), \\ \mathbb{E}[r^2] &= \left[\frac{1}{4} + \frac{|\langle s | U | s \rangle|^2}{4} + \frac{\text{Re}(\langle s | U | s \rangle)}{2} \right] + \left[\frac{1}{4} + \frac{|\langle s | U | s \rangle|^2}{4} - \frac{\text{Re}(\langle s | U | s \rangle)}{2} \right] = \frac{1}{2} + \frac{|\langle s | U | s \rangle|^2}{2}. \end{aligned}$$

Hence the variance is

$$\text{Var}(r) = \frac{1}{2} + \frac{|\langle s | U | s \rangle|^2}{2} - \text{Re}(\langle s | U | s \rangle)^2.$$

Let us compare to the variance of the ht protocol. Recall that the ht protocol uses a controlled- U operation denoted CU such that $CU|0\rangle|\psi\rangle = |0\rangle|\psi\rangle$ and $CU|1\rangle|\psi\rangle = |1\rangle U|\psi\rangle$. The computational basis measurement for the ancilla has probabilities,

$$\begin{aligned} P(x) &= \langle + | \langle s | CU^\dagger (H \otimes I^{\otimes n}) (|x\rangle\langle x| \otimes I^{\otimes n}) (H \otimes I^{\otimes n}) CU | + \rangle | s \rangle \\ &= \frac{1}{2} (\langle 0 | \langle s | + \langle 1 | \langle s | U^\dagger) (H \otimes I^{\otimes n}) (|x\rangle\langle x| \otimes I^{\otimes n}) (H \otimes I^{\otimes n}) (|0\rangle | s \rangle + |1\rangle U | s \rangle) \\ &= \frac{1}{4} (\langle 0 | \langle s | + \langle 1 | \langle s | + \langle 0 | \langle s | U^\dagger - \langle 1 | \langle s | U^\dagger) (|x\rangle\langle x| \otimes I^{\otimes n}) (|0\rangle | s \rangle + |1\rangle | s \rangle + |0\rangle U | s \rangle - |1\rangle U | s \rangle) \\ &= \frac{1}{4} (|\langle 0 | x \rangle|^2 (2 + \langle s | U^\dagger | s \rangle + \langle s | U | s \rangle) + |\langle 1 | x \rangle|^2 (2 - \langle s | U^\dagger | s \rangle - \langle s | U | s \rangle)) \\ &= \frac{1}{2} (1 + (-1)^x \text{Re}(\langle s | U | s \rangle)). \end{aligned}$$

We now define the binary random variable $r' = (-1)^x$ which is estimated by measuring the ancilla and discarding the rest. We have that

$$\mathbb{E}[r'] = \text{Re}(\langle s | U | s \rangle) \quad \mathbb{E}[(r')^2] = 1 \quad \text{Var}(r') = 1 - \text{Re}(\langle s | U | s \rangle)^2.$$

Therefore,

$$\text{Var}(r) = \text{Var}(r') - \frac{1 - |\langle s | U | s \rangle|^2}{2} \leq \text{Var}(r'),$$

with equality only when $|\langle s | U | s \rangle|^2 = 1$. An identical result is obtained when estimating the imaginary part (this corresponds to inserting an extra $S^\dagger$ gate into the circuit). So cf ev is usually better than ht in terms of variance.

Let us now consider the case of estimating $\langle s | U | s \rangle$ by combining the estimator of the real part, r , with the one for the imaginary part, r_c . For cf ev we have

$$\begin{aligned} \mathbb{E}[r + ir_c] &= \langle s | U | s \rangle \\ \text{Var}(r + ir_c) &= \text{Var}(r) + \text{Var}(r_c) = 1 + |\langle s | U | s \rangle|^2 - \text{Re}(\langle s | U | s \rangle)^2 - \text{Im}(\langle s | U | s \rangle)^2 = 1. \end{aligned}$$

Remarkably the variance of cf ev does not depend on the state $|s\rangle$. This is not the case for ht since

$$\begin{aligned}\mathbb{E}[r' + ir'_c] &= \langle s | U | s \rangle \\ \text{Var}(r' + ir'_c) &= \text{Var}(r') + \text{Var}(ir'_c) = 2 - \text{Re}(\langle s | U | s \rangle)^2 - \text{Im}(\langle s | U | s \rangle)^2 = 2 - |\langle s | U | s \rangle|^2,\end{aligned}$$

so we have

$$\text{Var}(r + ir_c) = \text{Var}(r' + ir'_c) - (1 - |\langle s | U | s \rangle|^2) \leq \text{Var}(r' + ir'_c),$$

with equality only when $|\langle s | U | s \rangle|^2 = 1$. cf ev can have half the variance of ht when $\langle s | U | s \rangle = 0$.

Finally, we comment on weighted trace estimation, which we use to estimate the Jones polynomial of Markov-closed braids. In this case, the estimator comprises an additional average over different realizations of the initial state $|s\rangle$ each with probability $P(s)$. Let us denote the s -dependent random variables as $r_s + ir_{c,s}$ and $r'_s + ir'_{c,s}$ for cf ev and ht , respectively. We have

$$\mathbb{E}[r_s + ir_{c,s}] = \mathbb{E}[\mathbb{E}[r_s + ir_{c,s} | s]] = \mathbb{E}[\langle s | U | s \rangle] = \text{tr} \left[\sum_s P(s) |s\rangle \langle s| U \right],$$

where we use the law of total expectation. This is the trace of U weighted by $P(s)$. The result is identical for the ht protocol, $\mathbb{E}[r'_s + ir'_{c,s}]$. Instead, for the variance we have

$$\begin{aligned}\text{Var}(r_s + ir_{c,s}) &= \mathbb{E}[\text{Var}(r_s + ir_{c,s} | s)] + \text{Var}(\mathbb{E}[r_s + ir_{c,s} | s]) = 1 + \mathbb{E}[|\langle s | U | s \rangle|^2] - |\mathbb{E}[\langle s | U | s \rangle]|^2, \\ \text{Var}(r'_s + ir'_{c,s}) &= \mathbb{E}[\text{Var}(r'_s + ir'_{c,s} | s)] + \text{Var}(\mathbb{E}[r'_s + ir'_{c,s} | s]) = 2 - |\mathbb{E}[\langle s | U | s \rangle]|^2,\end{aligned}$$

where we use the law of total variance. Once again we observe that the variance of cf ev is upper bounded by that of ht .

APPENDIX B: WEIGHTED SAMPLING FROM $\mathcal{F}$

Consider $|s\rangle \in \mathcal{F}_n$ as defined in Eq. (2). Since $s_0 = 0$, we must have $s_1 = 1$, and that $s_2 s_3 \dots s_{n-1} \in \mathcal{F}'_{n-2}$, where we define the set $\mathcal{F}'_n$ of Fibonacci strings of length n as

$$\mathcal{F}'_n = \{s \in \mathbb{B}^n \mid s_i + s_{i+1} > 0\}$$

and indeed we have that $\{s_2 s_3 \dots s_{n-1} \mid s \in \mathcal{F}\} = \mathcal{F}'_{n-2}$. Suppose we want to uniformly sample from $\mathcal{F}$ conditioned on the fact that $s_{n-1} = 0$. Then we must have $s_{n-2} = 1$, and the remaining substrings $s_2 \dots s_{n-3}$ satisfy $\{s_2 s_3 \dots s_{n-3} \mid s \in \mathcal{F}, s_{n-1} = 0\} = \mathcal{F}'_{n-4}$. Likewise if we condition on $s_{n-1} = 1$, then we have $\{s_2 s_3 \dots s_{n-2} \mid s \in \mathcal{F}, s_{n-1} = 1\} = \mathcal{F}'_{n-3}$. In either case, this is reduced to sampling uniformly from $\mathcal{F}'_m$ for some m . Zeckendorf's theorem states that there is a bijection between the integers $0 \leq k < f_{n+2}$ and the elements of $\mathcal{F}'_n$, so $|\mathcal{F}'_n| = f_{n+2}$. The bijection can be computed in linear time. Define a sequence k_i with $k_{n-1} = k$, then

$$k_{i-1} = \begin{cases} k_i & \text{if } k_i < f_{i+2} \\ k_i - f_{i+2} & \text{if } k_i \geq f_{i+2} \end{cases}, \quad s_i = \begin{cases} 1 & \text{if } k_i < f_{i+2} \\ 0 & \text{if } k_i \geq f_{i+2} \end{cases}.$$

Now consider sampling from $\mathcal{F}$ weighted by $\kappa(s) = \phi^{s_{n-1}}$. Since $|\mathcal{F}'_{n-3}| = f_{n-1}$ and $|\mathcal{F}'_{n-4}| = f_{n-2}$, the probability of

obtaining a given s_{n-1} should be

$$\begin{aligned}P(s_{n-1} = 1) &= \frac{\phi f_{n-1}}{\phi f_{n-1} + f_{n-2}}, \\ P(s_{n-1} = 0) &= \frac{f_{n-2}}{\phi f_{n-1} + f_{n-2}}.\end{aligned}$$

Then given s_{n-1} , the problem reduces to sampling uniformly from some $\mathcal{F}'_m$, which can be done by picking an integer $0 \leq k < f_{n+2}$ and computing s_i as described above. The overall method (see Algorithm 1) is as follows:

- (1) By definition $s_0 = 0$ and $s_1 = 1$. To obtain s_{n-1} sample according to $P(s_{n-1})$.
- (2) If $s_{n-1} = 0$ then $s_{n-2} = 1$ by definition. Pick $0 \leq k < f_{n-2}$ uniformly and use the bijection to fill in $s_2 \dots s_{n-3}$.
- (3) If $s_{n-1} = 1$ then pick $0 \leq k < f_{n-1}$ uniformly and use the bijection to fill in $s_2 \dots s_{n-2}$.

We can see that sampling from $\mathcal{F}$ weighted by $\kappa(s)$ is the same as sampling from the distribution

$$P(s) = \begin{cases} 0 & \text{if } |s\rangle \notin \mathcal{F} \\ \frac{\phi^{s_{n-1}}}{\phi^{n-1}} & \text{otherwise} \end{cases},$$

since the total weight of $\kappa(s)$ over $\mathcal{F}$ is given by $\phi f_{n-1} + f_{n-2} = \phi^{n-1}$.

ALGORITHM 1. A procedure based on Zeckendorf's theorem to efficiently classically sample a bit string $|s\rangle$ from $\mathcal{F}_n$ according to distribution $P(s)$.

```

1  $s_0 \leftarrow 0$ ;
2  $s_1 \leftarrow 1$ ;
3  $r \leftarrow$  sample uniformly in  $[0, 1]$ ;
4 if  $r \leq \frac{f_{n-1}}{\phi^{n-2}}$  then
5    $s_{n-1} \leftarrow 1$ ;
6    $j \leftarrow n - 2$ ;
7    $k \leftarrow$  sample random integer in  $[0, f_{n-1}]$ ;
8 else
9    $s_{n-1} \leftarrow 0$ ;
10   $s_{n-2} \leftarrow 1$ ;
11   $j \leftarrow n - 3$ ;
12   $k \leftarrow$  sample random integer in  $[0, f_{n-2}]$ ;
13 while  $j \geq 2$  do
14   if  $k \geq f_j$  then
15      $s_j \leftarrow 0$ ;
16      $k \leftarrow k - f_j$ ;
17   else
18      $s_j \leftarrow 1$ ;
19    $j \leftarrow j - 1$ ;
20 return  $|s\rangle$ 

```

APPENDIX C: NON-FIBONACCI ERROR DETECTION AND THE CONJUGATE TRICK

Suppose that $x_1, \dots, x_n$ is a sample from the circuit in Fig. 5(a). Without loss of generality let us assume that $\theta_B = 0$ and $b = 0$. According to Eq. (A1) case (3) has zero probability of occurring. That is, it should never be the case that $x_1 \neq 0$ or $(x_i \oplus s_i) + (x_{i+1} \oplus s_{i+1}) = 0$ for some $i \geq 3$. However, due to gate noise in the NISQ computer, we may actually observe case (3) implying that at least one hardware error has occurred. In this case the postprocessing function defined after Eq. (A1) will take value $g_2 = 0$ on the erroneous samples. We now argue that under reasonable noise assumptions these samples can be discarded during the estimation of expectation values.

From the definitions of r, g_1 and g_2 it can be verified that

$$\mathbb{E}[r] = \mathbb{E}[\mathbb{E}[r | g_2]] = P(g_2 = 1)\mathbb{E}[r | g_2 = 1]$$

Estimating r from the retained samples ($g_2 = 1$) is then equivalent to rescaling the original estimator by a factor of $1/P(g_2 = 1)$, without affecting the phase. Certain noise models, such as the depolarizing noise, have the effect of shrinking the magnitude of any expectation value toward zero. Under these noise assumptions we can safely discard any sample for which $g_2 = 0$. This will reduce the bias at the cost of an increase in variance. We expect the same to be at least approximately true for any nonadversarial noise model. This effect can be observed in our experiments by

comparing the green and orange markers in Fig. 6. We call this technique the *non-Fibonacci error detection*.

Next we discuss the *conjugate trick*. Suppose we estimated $\mathbb{E}_s[e^{i\theta} \langle s | U_B | s \rangle]$ for braid B from samples, obtaining the value $e^{i\theta} R$. For the conjugate braid B^* we have that $U_{B^*} = U_B^*$, since the braid generators satisfy $U_{\sigma^{-1}} = U_{\sigma}^\dagger = U_{\sigma}^*$ due to symmetry. Let us then estimate $\mathbb{E}_s[e^{i\theta} \langle s | U_{B^*} | s \rangle] = \mathbb{E}_s[e^{i\theta} \langle s | U_B^* | s \rangle]$ from samples as well, obtaining $e^{i\theta} R^*$. We can now recover the value of R up to a choice of signs as

$$R_{1\dots 4} = \pm \frac{1}{2} |e^{i\theta} R + e^{i\theta} R^*| \pm \frac{i}{2} |e^{i\theta} R - e^{i\theta} R^*|,$$

where each R_i correspond to a different choice of signs. To disambiguate, we proceed as follows. We can eliminate two of the four options by additionally calculating

$$R'_{1,2} = \pm |e^{i\theta} R| \sqrt{\frac{e^{i\theta} R}{e^{i\theta} R^*}}$$

and discarding the antipodal pair of $R_{1\dots 4}$ that is furthest from $R'_{1,2}$. The reason for not using $R'_{1,2}$ initially is that it is not obvious that the variance of R'_i is the same as for R_i . The remaining values of R_i are positioned at antipodal points. By selecting the value nearest to $e^{i\theta} R$, we ensure the signs are chosen correctly, provided that $|\theta| < \pi/2$.

Applying the conjugate trick relies on the assumption that the overall phase error $e^{i\theta}$ is the same for every measurement shot, for both B and B^* . This may not be the case in practice—for example, if each gate accumulated phase error at a fixed rate, then even slight changes to this rate (for instance due to hardware calibration) could cause large changes in the overall phase error θ . Suppose that the phase error is consistent across shots, but for B^* it is instead given by $e^{i\theta'}$ with $\theta' \neq \theta$. Then we can see that the value recovered by the conjugate trick is given by

$$\begin{aligned} \tilde{R}_{1\dots 4} &= \pm \frac{1}{2} |e^{i\theta} R + e^{i\theta'} R^*| \pm \frac{i}{2} |e^{i\theta} R - e^{i\theta'} R^*| \\ &= \pm \frac{1}{2} |e^{i\frac{\theta+\theta'}{2}} e^{i\frac{\theta-\theta'}{2}} R + e^{i\frac{\theta+\theta'}{2}} e^{-i\frac{\theta-\theta'}{2}} R^*| \\ &\quad \pm \frac{i}{2} |e^{i\frac{\theta+\theta'}{2}} e^{i\frac{\theta-\theta'}{2}} R - e^{i\frac{\theta+\theta'}{2}} e^{-i\frac{\theta-\theta'}{2}} R^*| \\ &= \pm \frac{1}{2} |e^{i\frac{\theta+\theta'}{2}} (e^{i\frac{\theta-\theta'}{2}} R) + e^{i\frac{\theta+\theta'}{2}} (e^{i\frac{\theta-\theta'}{2}} R^*)| \\ &\quad \pm \frac{i}{2} |e^{i\frac{\theta+\theta'}{2}} (e^{i\frac{\theta-\theta'}{2}} R) - e^{i\frac{\theta+\theta'}{2}} (e^{i\frac{\theta-\theta'}{2}} R^*)|. \end{aligned}$$

So we can see that this is the same as the case where $\theta = \theta'$, except with $R \rightarrow e^{i((\theta-\theta')/2)} R$ and $\theta \rightarrow (\theta + \theta')/2$. This will be disambiguated to $\tilde{R} = e^{i((\theta-\theta')/2)} R$ if $|(\theta + \theta')/2| < \pi/2$ by the same argument as above. Then, in the absence

of finite sampling noise, the relative errors of the estimates with and without the conjugate trick are given by

$$\varepsilon_{\text{with}} = \frac{|R - \tilde{R}|}{|R|} = \left| 1 - e^{i\frac{\theta - \theta'}{2}} \right| = 2 \left| \sin \left(\frac{\theta - \theta'}{4} \right) \right|,$$

$$\varepsilon_{\text{without}} = \frac{|R - e^{i\theta} R|}{|R|} = |1 - e^{i\theta}| = 2 \left| \sin \left(\frac{\theta}{2} \right) \right|.$$

So applying the conjugate trick will be beneficial compared to the original estimate whenever $|\theta - \theta'| \leq 2|\theta|$, i.e., the relative error between θ and θ' must be at most 200%.

Conversely, in the worst case, suppose that the phase error of each shot is uniformly random, then we have

$$\mathbb{E}[r] = \mathbb{E}[\mathbb{E}[r | \theta]] = \mathbb{E}[e^{i\theta}] \mathbb{E}[\langle s | U_B | s \rangle] = 0,$$

so the conjugate trick cannot recover any information. Therefore, we will now consider the case where the change in phase error is roughly zero within a single measurement shot, but may be arbitrarily different between measurement shots. We will assume that we are estimating the amplitude of a single bitstring $\langle s | U_B | s \rangle$, and extend this to the case of trace estimation later. We call this the *shot-level conjugate trick*. Consider batching together four different measurements into a single circuit: the real and imaginary parts of R , and the real and imaginary parts of R^* . We assume that they all have the same phase error because they are within the same circuit (which is not too large). For measurement shot i , define the following random variables, corresponding to the measurement outcomes of each of the four parts respectively,

$$X_i \sim \text{EV}(x, y, \theta_i) \quad Y_i \sim \text{EV}(-y, x, \theta_i)$$

$$X_i^* \sim \text{EV}(x, -y, \theta_i) \quad Y_i^* \sim \text{EV}(y, x, \theta_i),$$

where $x + iy = \langle s | U_B | s \rangle$ and θ_i is the phase error for this measurement shot. $\text{EV}(x, y, \theta)$ is the distribution of measurement outcomes given by `cfev` estimating a quantity with real part x , imaginary part y , and phase error θ , so that

$$P(X_i = 0) = \frac{1 - x^2 - y^2}{2},$$

$$P(X_i = +1) = \frac{1 + x^2 + y^2}{4} + \frac{x \cos(\theta_i) - y \sin(\theta_i)}{2},$$

$$P(X_i = -1) = \frac{1 + x^2 + y^2}{4} - \frac{x \cos(\theta_i) - y \sin(\theta_i)}{2},$$

as computed in Appendix A. Then a straightforward but tedious calculation shows that

$$\mathbb{E}[X_i X_i^* + Y_i Y_i^*] = x^2 - y^2$$

$$\mathbb{E} \left[\frac{X_i^2 + Y_i^2 + (X_i^*)^2 + (Y_i^*)^2}{2} - 1 \right] = x^2 + y^2,$$

which is independent of θ_i , and hence we can recover x^2 and y^2 as

$$X_i^{EC} = \frac{(X_i + X_i^*)^2 + (Y_i + Y_i^*)^2 - 2}{4} \quad \mathbb{E}[X_i^{EC}] = x^2,$$

$$Y_i^{EC} = \frac{(X_i - X_i^*)^2 + (Y_i - Y_i^*)^2 - 2}{4} \quad \mathbb{E}[Y_i^{EC}] = y^2.$$

Then taking N shots, even if each θ_i is arbitrary, we have that

$$\pm \sqrt{\mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N X_i^{EC} \right]} = \pm \sqrt{\frac{1}{N} \sum_{i=1}^N \mathbb{E}[X_i^{EC}]}$$

$$= \pm \sqrt{\frac{1}{N} \sum_{i=1}^N x^2} = \pm x$$

and likewise for Y_i^{EC} . Let $M_N = 1/N \sum_{i=1}^N X_i^{EC}$, then because $-\frac{1}{2} \leq X_i^{EC} \leq \frac{3}{2}$ we have

$$P(|M_N - x^2| \geq t) \leq 2e^{-N \frac{t^2}{2}}$$

from Hoeffding's inequality, and thus,

$$P \left(\left| \sqrt{M_N} - |x| \right| \geq t \right) = P \left(|M_N - x^2| \geq t \left| \sqrt{M_N} + |x| \right| \right)$$

$$\leq P \left(|M_N - x^2| \geq t|x| \right) \leq 2e^{-N \frac{t^2 x^2}{2}}.$$

Note that this inequality can be tightened further [59], but this is enough to imply that the following is a consistent estimator for $x + iy$ up to the sign of its components

$$R^{EC} = \pm \sqrt{\frac{1}{N} \sum_{i=1}^N X_i^{EC}} \pm i \sqrt{\frac{1}{N} \sum_{i=1}^N Y_i^{EC}}.$$

Since the square root function is not Lipschitz continuous, the variance will depend inversely on the value of $|x + iy|$, and in any case will be substantially larger than the variance from the regular conjugate trick.

To adapt this to trace estimation, if we let x and y vary with each shot such that $\mathbb{E}[x] = x'$ and $\mathbb{E}[y] = y'$ then this estimator will be biased, since $\mathbb{E}[x^2] \neq (x')^2$ in general. If the variance of x is small, this may be a reasonable estimator, but to obtain a consistent estimator for the trace, one can instead estimate R^{EC} for each bit string $|s\rangle$ individually and combine these. However, it is important to note that this can only be unbiased (up to a choice of signs) when the signs of the components of each amplitude on the diagonal are all positive or all negative, since the sign information cannot be recovered from the shot-level trick.

We will now prove that this works for estimating the absolute value of the real part of the trace. The imaginary part follows similarly. Let $\{s_j\}$ for $1 \leq j \leq m$ and s be independent identically distributed Fibonacci strings according to $s \sim p(s)$. Then we will estimate $q(s_j) = |\text{Re}(\langle s_j | U_B | s_j \rangle)|$ using the shot-level conjugate trick with N shots, and average over these. Our estimator is thus defined as $Z = 1/m \sum_{j=1}^m \sqrt{M_N(s_j)}$ where $M_N(s_j) =$

$1/N \sum_{i=1}^N X_i^{EC}(s_j)$ is defined above. Furthermore, we can define B as follows, which is the expected bias due to the sign information of each amplitude being unavailable,

$$B = \left| |\text{Re}(\mathbb{E}[\langle s | U_B | s \rangle])| - \mathbb{E}[q(s)] \right|.$$

Using the stronger concentration inequality of [59], we bound the conditional probability

$$P\left(\left|\sqrt{M_N(s_j)} - q(s_j)\right| \geq t \mid s_j\right) \leq \begin{cases} 2e^{-\frac{N}{2}(q(s_j)^2 - (q(s_j) - t)^2)} & \text{if } t < q(s_j) \\ 2e^{-\frac{N}{2}(q(s_j)^2 - (q(s_j) + t)^2)} & \text{if } t \geq q(s_j) \end{cases} \leq 2e^{-\frac{N}{2}t^4}.$$

Applying the tail-sum formula for expected values, we find

$$\mathbb{E}\left[\left|\sqrt{M_N(s_j)} - q(s_j)\right| \mid s_j\right] = \int_0^\infty P\left(\left|\sqrt{M_N(s_j)} - q(s_j)\right| \geq t \mid s_j\right) dt \leq \int_0^\infty 2e^{-\frac{N}{2}t^4} dt = \frac{2^{\frac{5}{4}} \Gamma(\frac{5}{4})}{N^{\frac{1}{4}}} \leq \frac{3}{N^{\frac{1}{4}}}.$$

First we will show that Z is a consistent estimator for $\mathbb{E}[q(s)]$,

$$\begin{aligned} |\mathbb{E}[Z - \mathbb{E}[q(s)]]| &= \left| \mathbb{E}\left[\frac{1}{m} \sum_{j=1}^m \left(\sqrt{M_N(s_j)} - \mathbb{E}[q(s)]\right)\right] \right| \\ &= \left| \frac{1}{m} \sum_{j=1}^m \mathbb{E}\left[\sqrt{M_N(s_1)} - \mathbb{E}[q(s)]\right] \right| = \left| \mathbb{E}\left[\sqrt{M_N(s_1)} - \mathbb{E}[q(s)]\right] \right| \\ &= \left| \mathbb{E}\left[\sqrt{M_N(s_1)} - q(s_1)\right] \right| = \left| \mathbb{E}\left[\sqrt{M_N(s_1)} - q(s_1)\right] \right| \leq \mathbb{E}\left[\left|\sqrt{M_N(s_1)} - q(s_1)\right|\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[\left|\sqrt{M_N(s_1)} - q(s_1)\right| \mid s_1\right]\right] \leq \mathbb{E}\left[\frac{3}{N^{\frac{1}{4}}}\right] = \frac{3}{N^{\frac{1}{4}}}. \end{aligned}$$

And so from the triangle inequality, we get a bound on the bias of Z as an estimator for $|\text{Re}(\mathbb{E}[\langle s | U_B | s \rangle])|$,

$$\begin{aligned} |\text{Bias}| &= |\mathbb{E}[Z - |\text{Re}(\mathbb{E}[\langle s | U_B | s \rangle])|]| \\ &\leq |\mathbb{E}[Z - \mathbb{E}[q(s)]]| + B \leq \frac{3}{N^{\frac{1}{4}}} + B. \end{aligned}$$

Thus, in the situation where $B = 0$, Z is a consistent estimator, with bias that goes like $O(N^{-1/4})$. On the other hand, because each $M_N(s_j)$ is bounded, we would expect that the standard deviation of Z is $O(m^{-1/2})$ from Hoeffding's inequality. This characterizes the bias-variance trade-off. Now, imagine that we have $B = 0$ and a fixed budget of $k = Nm$ shots, then we can pick $N = k^{2/3}$ and $m = k^{1/3}$ so that the bias and standard deviation both scale as $O(k^{-1/6})$. Note that this is much slower than standard Monte Carlo convergence, but we would expect that it performs much better than this in practice, since the bias of Z

will converge asymptotically faster than $O(N^{-1/4})$ whenever $\mathbb{E}_s[|\text{Re}(\langle s | U_B | s \rangle)|]$ is bounded away from zero by a constant.

APPENDIX D: COMPILATION PIPELINE

1. Braid augmentation

To generate braids for which the value of the Jones polynomial is known in advance, we take the following approach. Let $B = \otimes_{i=1}^k b_i$ with $3k$ strands be the tensor product of 3-strand braids b_i . The Jones polynomial for each $M(b_i)$ can be calculated trivially, and the Jones polynomial of $M(B)$ is given as

$$V_{M(B)}\left(e^{i\frac{2\pi}{5}}\right) = \phi^{k-1} \prod_{i=1}^k V_{M(b_i)}\left(e^{i\frac{2\pi}{5}}\right),$$

so it can be computed efficiently classically. Then, we generate random braids A and A^{-1} such that $AA^{-1} = I$ is the identity braid on $3k$ strands. Let $B' = A^{-1}BA$, then since the Markov closure of a braid is unchanged by conjugation, we have $V_{M(B')}(e^{i(2\pi/5)}) = V_{M(B)}(e^{i(2\pi/5)})$ so this can also be computed efficiently. By varying the number of crossings in A and the number of strands in B , the size of B' can be controlled.

First, we describe how to generate B . We want the dominant error of $\text{cf} \circ \text{ev}$ when applied to B' to be ϵ_{noise} (which is what we wish to characterize) rather than ϵ_{shot} , so we must ensure that the value of $|\mathbb{E}_s[\langle s | U_B | s \rangle]|$ does not concentrate exponentially around zero. To this end, we first enumerate all 3-strand braids b_j up to a given number of generators (we used four in practice), and calculated $T_j = \log |\mathbb{E}_s[\langle s | U_{b_j} | s \rangle]|$ for each. Then, suppose that we want to construct braid B on $3k$ strands. Each b_i is sampled independently and identically from this set $\{b_j\}$ such that

$$|\mathbb{E}_s[\langle s | U_B | s \rangle]| = \prod_{i=1}^k |\mathbb{E}_s[\langle s | U_{b_i} | s \rangle]| = \exp \left[\sum_{i=1}^k T_i \right]$$

is approximately uniformly distributed in $[0, 1]$. Note that it is not possible for this to be exactly uniformly distributed for any particular k because the set $\{T_j\}$ is finite. We construct D_k as

$$P(D_k = b_i) = \frac{P(D'_k = T_i)}{|\{b_j \mid T_j = T_i\}|},$$

where D'_k is a distribution over the set $\{T_j\}$. Clearly $T_i \sim D'_k$, and b_i is selected uniformly at random from all b_j such that $T_j = T_i$. Thus, the condition that $|\mathbb{E}_s[\langle s | U_B | s \rangle]|$ be uniformly distributed depends only on D'_k . Let $t_1, t_2, \dots, t_n$ be the unique values of T_j , and define $P(D'_k = t_j) = p_j$.

Next we use the idea of a moment generating function, which for a random variable R is defined in terms of a free variable x as $\mathbb{E}[e^{xR}]$. It is defined this way because when expressed as a formal power series in x , the coefficient of x^k is $m_k/k!$ where m_k is the k th raw moment of R . Let $Z = \log |\mathbb{E}_s[\langle s | U_B | s \rangle]| = \sum_{i=1}^k T_i$ so that $|\mathbb{E}_s[\langle s | U_B | s \rangle]| \sim e^Z$. The moment generating function of Z can be calculated as

$$\mathbb{E}[e^{xZ}] = \mathbb{E}[e^{x \sum_{i=1}^k T_i}] = \prod_{i=1}^k \mathbb{E}[e^{xT_i}] = \left(\sum_{j=1}^n p_j e^{xT_j} \right)^k.$$

To enforce that $|\mathbb{E}_s[\langle s | U_B | s \rangle]| \sim e^Z$ is approximately uniform, we can pick p_j such that the moment generating function of Z matches the desired distribution. In particular, let $U \sim \mathcal{U}([0, 1])$ then we want e^Z to match U , and

hence Z to match $\log U$. Since we have

$$\mathbb{E}[e^{x \log(U)}] = \mathbb{E}[U^x] = \int_0^1 u^x du = \frac{1}{x+1},$$

then matching this for Z yields

$$\begin{aligned} \mathbb{E}[e^{xZ}] &= \left(\sum_{j=1}^n p_j e^{xT_j} \right)^k \approx \frac{1}{x+1} \\ \Rightarrow \sum_{j=1}^n p_j e^{xT_j} &\approx \frac{1}{\sqrt[k]{x+1}}. \end{aligned}$$

By picking a finite set of x values where we want this to be satisfied (we used 100 equally spaced points in $0 \leq x \leq \frac{15}{2}$), this is transformed into a regression problem subject to the constraint that $p_j \geq 0$ and $\sum_j p_j = 1$, which can be solved by any standard constrained convex optimization method.

In theory, any method of generating random braids could be used to construct A , while A^{-1} is given by reversing the order and sign of all the crossings in A . However, with this method, A and A^{-1} will be structurally very similar, and the overall circuit will be quite similar to mirror benchmarking techniques. Therefore, to generate a greater variety of circuit structures, and thus a more comprehensive benchmark, we use a method based on sorting networks.

First, we generate a random network A of crossings on n strands, not yet committing to whether each crossing is under or over. This is done in a brick-wall layout (see Fig. 13) by picking randomly whether each “brick” is a crossing or the identity. The number of layers in the brick-wall can be varied to control the amount of crossings in the final braid. Regardless of the sign of each crossing, this induces the same permutation π on the strands themselves, considering the group homomorphism f from the braid

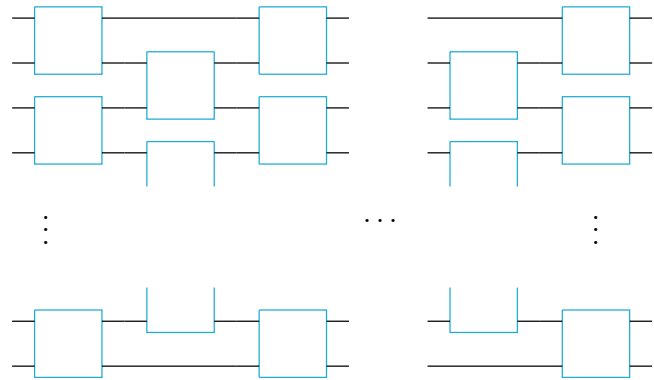
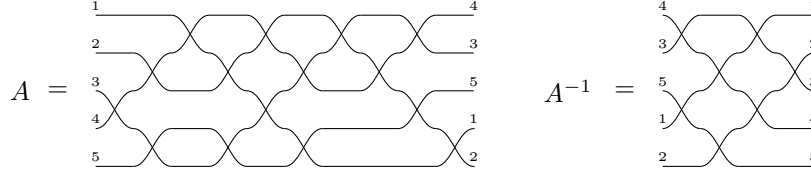


FIG. 13. Random braid generated on a brick-wall pattern, where “bricks” are chosen from the set $\{\sigma, \sigma^{-1}, 1_2\}$ containing the braid generators and the 2-strand identity according to a strategy described in Appendix D 1.

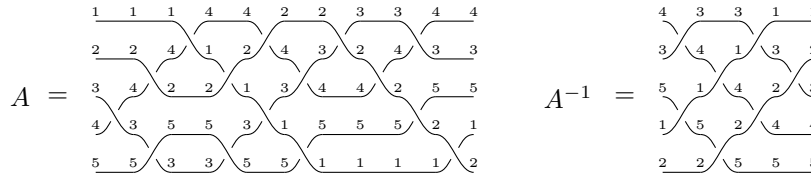
group B_n to the symmetric group S_n that maps crossings to transpositions $f : \sigma_i^{\pm 1} \rightarrow (i \ i+1)$. Now we can generate a similar network A^{-1} which induces the permutation π^{-1} via any deterministic method of decomposing a permutation into transpositions of consecutive pairs, for instance a

sorting network. We used the odd-even sorting network to produce a brick-wall style network with depth at most n . For instance, see the following example with $n = 5$ where A is generated using a 10-layer brick-wall,



Here, the small numbers on each strand indicate the induced permutation. Clearly, the concatenation of these two would induce the identity permutation as desired. Because of this structure, A^{-1} will in general be much shallower than A if the number of layers in the brick-wall is much larger than n . We need to pick a sign for each crossing so that AA^{-1} is indeed the identity braid on n strands. Note that not all sets of signs work, in general we might end up with some braid from the pure braid group $P_n \cong \ker f$ (that is, braids whose induced

permutation is the identity). Thus, we use the following simple method which guarantees success, but could perhaps be improved in the future: we choose at random a total ordering on the strands given by $x \prec y$, and for every crossing $\sigma_i^{\pm 1}$ between strands i and $i+1$ we pick σ_i^{+1} if $\pi'(i+1) \prec \pi'(i)$ and σ_i^{-1} if $\pi'(i) \prec \pi'(i+1)$, where π' is the permutation induced by all the crossings up to this point. Continuing the example above, using the ordering $4 \prec 3 \prec 5 \prec 1 \prec 2$, we obtain



where the numbers after each layer indicate the induced permutation up to that point π' . To see that this construction will always generate A and A^{-1} such that AA^{-1} is the identity braid, we can use induction on the strands. Starting from the top-most strand, since it lies either fully atop or beneath every other strand, a series of “slide” and “poke” moves (cf. Fig. 14) can reduce the braid so that no generators act on this strand. This can be repeated for all strands in sequence until the identity is reached.

Lastly, we comment briefly on why this method was chosen—we previously considered two options: either constructing ABA^{-1} as above, but using a family of n -strand braids with a known Jones polynomial, in particular the torus knots, or taking a small braid for which the Jones polynomial of the closure is known or can be computed classically, and augmenting it with random Markov moves to form a large braid. In both cases, the value of the Jones polynomial does not scale exponentially with the

number of strands, and hence the expectation we would need to measure in `cfev` would decrease as $O(\phi^{-n})$. Thus ϵ_{shot} would quickly become the dominant error, which is unhelpful for this benchmark as it is intended to measure ϵ_{noise} .

2. Braid simplification

Given a specific knot or link that you might wish to evaluate with `cfev` it is useful to obtain a braid whose closure is the given link, called a *braid representative*, which minimizes the number of crossings so as to minimize both the time required to run the algorithm and the ϵ_{noise} error incurred. A braid representative can be efficiently obtained via Vogel’s algorithm [60], but this may have many crossings. Note that identifying a small braid representative is hard in general. For example minimizing the number of strands is hard [61], minimizing the

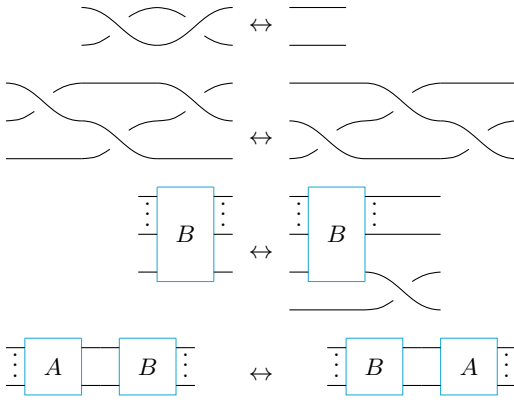


FIG. 14. Braid rewrites that relate topologically equivalent braids upon Markov closure. From top to bottom we have the local moves: Reidemeister II or “poke” ($\sigma_i \sigma_i^{-1} = 1_2$), Reidemeister III or “slide” ($\sigma_i \sigma_{i+1} \sigma_i \leftrightarrow \sigma_{i+1} \sigma_i \sigma_{i+1}$), “stabilization” ($B \leftrightarrow B \sigma_{n+1}$), and the global move “cycle” or “conjugation” ($AB \leftrightarrow BA$), where A, B are arbitrary n -strand braids.

number of crossings is NP-hard [62], and when minimizing both strands and crossings this is a complete knot invariant and hence even harder [63]. Therefore, in our pipeline [52] we include a heuristic solution to this problem based on applying peephole optimization to a given braid. Using Berger’s algorithm for minimizing 3-strand braids [64], we rewrite each 3-strand sub-braid and apply random “slide” and conjugation moves until no further improvement is obtained. This is fast in practice but may lead to larger numbers of crossings than other slower heuristic methods. Anecdotally, we observed that our method is $\sim 20\%$ worse than the physics-based heuristic proposed in Ref. [55] for uniformly random 10-strand braid words.

3. Braid generator circuits

For U_σ^1 , the parameters for the native 1- and 2-qubit gates in Fig. 4(b) are as follows:

$$\begin{aligned} \alpha_0 &= a_1 & \beta_0 &= 0 & \chi_0 &= -b_1 & \theta_0 &= -\frac{\pi}{10}, \\ \alpha_1 &= -a_1 & \beta_1 &= -b_1 & \chi_1 &= \frac{\pi}{2} & \theta_1 &= 2b_1 + \frac{3\pi}{10}, \\ \alpha_2 &= -a_1 & \beta_2 &= -b_1 - \frac{3\pi}{10} & \chi_2 &= -b_1 & \theta_2 &= \frac{\pi}{5}, \\ \alpha_3 &= a_1 & \beta_3 &= -2b_1 - \frac{3\pi}{10}, \end{aligned}$$

where the angles a_1 and b_1 are defined by

$$a_1 = \sin^{-1} \left(\sqrt{\frac{4}{3\phi}} \right) - \pi \quad b_1 = \sin^{-1} \left(\phi \sqrt{\frac{3}{8}} \right)$$

and the eigenvalue corresponding to $|000\rangle$ is $e^{i\alpha} = e^{i(2\pi/5)}$. Similarly for U_σ^2 we have

$$\begin{aligned} \alpha_0 &= \pi - a_2 & \beta_0 &= 0 & \chi_0 &= -b_2 & \theta_0 &= \frac{\pi}{5}, \\ \alpha_1 &= a_2 & \beta_1 &= -b_2 & \chi_1 &= \frac{3\pi}{5} & \theta_1 &= 2b_2 + \frac{3\pi}{5}, \\ \alpha_2 &= -a_2 & \beta_2 &= \frac{2\pi}{5} - b_2 & \chi_2 &= -b_2 & \theta_2 &= -\frac{2\pi}{5}, \\ \alpha_3 &= \pi - a_2 & \beta_3 &= -2b_2 - \frac{3\pi}{5}, \end{aligned}$$

where the angles a_2 and b_2 are defined by

$$a_2 = \sin^{-1} \left(\frac{2}{\sqrt{3+2\phi}} \right) \quad b_2 = \sin^{-1} \left(\frac{\sqrt{3\phi-1}}{2} \right) - \pi$$

and the eigenvalue corresponding to $|000\rangle$ is $e^{i\alpha} = e^{-i(2\pi/5)}$. The parameters for U_σ^k with $k \geq 3$ have more complicated closed forms, so we calculated them numerically instead, and the values are listed in an online repository along with the pipeline code [52]. Note that these values are not the only possible expression of these unitaries in the form given in Fig. 4(b).

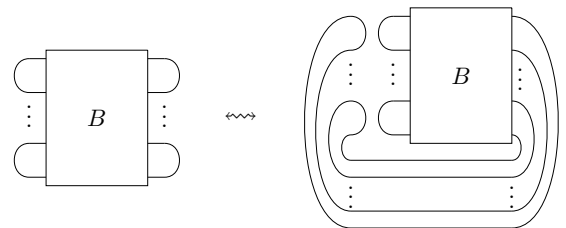
APPENDIX E: EVALUATING THE JONES FOR PLAT-CLOSED BRAIDS

Theorem E1. For any braid B with $n-1$ strands, where $n-1$ is even,

$$V_{P(B)} \left(e^{i\frac{2\pi}{5}} \right) = \left(-e^{-i\frac{3\pi}{5}} \right)^{3w_B} \phi^{\frac{n}{2}-\frac{3}{2}} \langle \alpha | U_B | \alpha \rangle,$$

where $|\alpha\rangle = |01010 \dots 10\rangle$, w_B is the writhe of B , and $P(B)$ is the link corresponding to the Plat closure of B .

Proof. We adapt the proof of [4, Theorem 3.2] that uses the path model representation of B_n to the Fibonacci string representation of [6] which we use in Sec. II. For any braid B , the following two link diagrams are ambient isotopic and hence have the same Jones polynomial:



Note that the left diagram corresponds to the $P(B)$ while the right corresponds to $M(B \cdot E)$, where $B \cdot E$ is a tangle

and E is an element of the Temperley-Lieb algebra,

$$E = \begin{array}{c} \cap \quad \cap \\ \vdots \quad \vdots \\ \cup \quad \cup \end{array} \quad B \cdot E = \begin{array}{c} \cap \quad \cap \\ \vdots \quad \vdots \\ \cup \quad \cup \end{array} \boxed{B}$$

Therefore, we have that $V_{P(B)}(x) = V_{M(B \cdot E)}(x)$. In Sec. II we give an expression for $V_{M(B')}(e^{i(2\pi/5)})$ for any braid B' in terms of a unitary representation $U_{B'}$, which was originally proven by Kauffman and Lomonaco [6, Theorem 2] (this is equivalent to [4, Lemma 2.2] in the path model representation). They show that this also holds when B' is any tangle (i.e., it contains Temperley-Lieb generators), although $U_{B'}$ may be nonunitary, and give the representation in this case. Specifically, if B has $n-1$ strands, we have that

$$U_{B \cdot E} = U_B U_E \quad U_E = \prod_{i=0}^{\frac{n-1}{2}-1} I_{2i} \otimes M \otimes I_{n-1-2(i+1)}$$

$$M = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \phi & & & \\ & & & 0 & & \\ & & & & 1 & \\ & & & & & a & b \\ & & & & & & 0 \\ & & & & & & & b & 1 \end{pmatrix},$$

where $a = 1/\phi$, $b = \sqrt{1 - (1/\phi^2)}$, and I_k is the identity matrix of dimension $2^k \times 2^k$ (note that $M = e^{i(3\pi/5)} U_{\sigma_1} + e^{i(\pi/5)} I_3$).

We will now show that $U_E |s\rangle = 0$ for all $|s\rangle \in \mathcal{F}$ except for $|\alpha\rangle$. Suppose that $U_E |s\rangle \neq 0$. Since the terms $I_{2i} \otimes M \otimes I_{n-1-2(i+1)}$ and $I_{2j} \otimes M \otimes I_{n-1-2(j+1)}$ commute for any i and j , we must have that $(I_{2i} \otimes M \otimes I_{n-1-2(i+1)}) |s\rangle \neq 0$ for all $0 \leq i < (n-1)/2$. If $s_{2i} = 0$, then $s_{2i+1} = 1$ by the definition of $\mathcal{F}$. In order for $(I_{2i} \otimes M \otimes I_{n-1-2(i+1)}) |s\rangle \neq 0$, we must have $s_{2(i+1)} = 0$ since

$$\begin{aligned} & (I_{2i} \otimes M \otimes I_{n-1-2(i+1)}) |s\rangle \\ &= |s_0 \cdots s_{2i-1}\rangle \otimes M |s_{2i} s_{2i+1} s_{2(i+1)}\rangle \otimes |s_{2i+3} \cdots s_{n-2}\rangle \\ &= |s_0 \cdots s_{2i-1}\rangle \otimes M |01 s_{2(i+1)}\rangle \otimes |s_{2i+3} \cdots s_{n-2}\rangle, \end{aligned}$$

but $M |011\rangle = 0$ and $M |010\rangle = \phi |010\rangle \neq 0$. From the definition of $\mathcal{F}$, we have $s_0 = 0$ and $s_1 = 1$, so by induction we must have $s_{2i} = 0$ and $s_{2i+1} = 1$ for all $0 \leq i < (n-1)/2$, and thus $|s\rangle = |\alpha\rangle$. In this case, $(I_{2i} \otimes M \otimes I_{n-1-2(i+1)}) |\alpha\rangle = \phi |\alpha\rangle$, so $U_E |\alpha\rangle = \phi^{(n-1)/2} |\alpha\rangle$. Therefore, we have that

$$\begin{aligned} V_{P(B)}(e^{i\frac{2\pi}{5}}) &= V_{M(B \cdot E)}(e^{i\frac{2\pi}{5}}) = \left(-e^{-i\frac{3\pi}{5}}\right)^{3w_B} \phi^{n-2} \mathbb{E}_s[\langle s | U_{B \cdot E} | s \rangle] \\ &= \left(-e^{-i\frac{3\pi}{5}}\right)^{3w_B} \phi^{n-2} \sum_{s \in \mathcal{F}} \frac{\phi^{s_{n-1}}}{\phi^{n-1}} \langle s | U_{B \cdot E} | s \rangle \\ &= \left(-e^{-i\frac{3\pi}{5}}\right)^{3w_B} \phi^{-1} \sum_{s \in \mathcal{F}} \phi^{s_{n-1}} \langle s | U_B U_E | s \rangle \\ &= \left(-e^{-i\frac{3\pi}{5}}\right)^{3w_B} \phi^{\frac{n}{2}-\frac{3}{2}} \langle \alpha | U_B | \alpha \rangle. \quad \blacksquare \end{aligned}$$

The quantum protocol presented in Sec. II can be used to solve the BQP-complete problem of additively approximating the Jones polynomial of Plat-closed braids at the fifth root of unity. For every circuit execution, we fix $|s\rangle = |01010 \cdots 10\rangle$ and estimate the desired quantity as $V_{P(B)}(e^{i(2\pi/5)}) = (-e^{-i(3\pi/5)})^{3w_B} \phi^{(n/2)-\frac{3}{2}} \langle \alpha | U_B | \alpha \rangle$.

We will briefly comment on why this definition appears different to that in the Aharonov-Jones-Landau (AJL) algorithm. In the AJL algorithm, their computation of the Jones polynomial for the Plat closure has the following form [4, Theorem 3.2] (where n in their notation is $n-1$ in our notation, since we count qubits rather than strands),

$$V_{P(B)}(e^{i\frac{2\pi}{k}}) = (-A)^{3w_B} \frac{d^{\frac{3}{2}(n-1)-1} \sin(\frac{\pi}{k})}{N} \langle \alpha | U_B | \alpha \rangle,$$

where A is a phase depending on k and N is an exponentially large factor defined by

$$N = \sum_{l=1}^{k-1} \sin\left(\frac{l\pi}{k}\right) \dim \mathcal{H}_{n-1,k,l}$$

and $\dim \mathcal{H}_{n-1,k,l}$ is the number of walks of length $n-1$ on the path graph of $k-1$ vertices that start on the left-most vertex $l=1$ and end on vertex l . Note that their definition of $|\alpha\rangle$ and U_B are slightly different than ours, although it is not important in this case. Substituting $k=5$, we have

$$d = \phi \quad A = e^{-i\frac{3\pi}{5}} \dim \mathcal{H}_{n-1,5,l} = \begin{cases} f_{n-2} & \text{if } l=1 \text{ and } n \text{ is odd or } l=4 \text{ and } n \text{ is even,} \\ f_{n-1} & \text{if } l=2 \text{ and } n \text{ is even or } l=3 \text{ and } n \text{ is odd,} \\ 0 & \text{otherwise,} \end{cases}$$

so since $\sin(x)$ is symmetric around $\pi/2$ and $n - 1$ is either even or odd we can compute N as

$$\frac{N}{\sin(\frac{\pi}{5})} = f_{n-2} + \frac{\sin(\frac{2\pi}{5})}{\sin(\frac{\pi}{5})} f_{n-1} = f_{n-2} + \phi f_{n-1} = \phi^{n-1}$$

and thus we get

$$V_{P(B)}\left(e^{i\frac{2\pi}{5}}\right) = (-A)^{3w_B} \phi^{\frac{n}{2} - \frac{3}{2}} \langle \alpha | U_B | \alpha \rangle,$$

which matches the theorem above, excepting the differences in U_B and $|\alpha\rangle$.

APPENDIX F: NUMERICAL EXPERIMENT DETAILS

1. Noisy circuit simulation

We simulated `cfev` as compiled to the native gate set of Quantinuum's H-series devices using a statevector method. Random Pauli errors were inserted after every U_{1q} 1- and R_{ZZ} 2-qubit gate to create a depolarizing channel. No errors were inserted for R_Z gates, since in practice these are implemented entirely in software. Random bit-flip initialization errors were applied on each qubit at the beginning of each circuit, and asymmetric bit-flip errors were applied to the measurement results. See the repository [52] for the exact error rates and parameters. Non-Fibonacci error detection was applied to the output of these simulations, but the conjugate trick was not used (since no coherent errors were included, it is unnecessary).

The simulations were run with a custom statevector simulator built using the Triton GPU programming environment [65]. This was necessary to avoid the performance penalty incurred in existing statevector simulators (for example, we saw up to 50x slow down relative to Qiskit-Aer [66] on the same hardware) when simulating circuits which must vary between shots (this is due to the bit-string $|s\rangle$ being randomly sampled, and hence V_{cat} must be recompiled for each shot). A Qiskit-Aer-based simulation is also included in the repository, which we cross-checked against our custom implementation to ensure correctness.

The benchmark set was generated using the technique from Sec. III, specifically according to the method in Appendix D 1. The number of layers in the initial brick-wall circuit was varied from 5 to 100 in steps of 5. The number of strands varied from 10 to 15 with: 10k braids on 10 strands, 5k each for 11 and 12 strands, 2.4k for 13 strands, 1.2k for 14 strands, and 640 braids on 15 strands. This was done to ensure that the time spent on each number of strands was roughly the same.

In Fig. 8 we see how the relative error depends on the number of crossings. We use a power-law fit to extrapolate because it provided a better fit than a linear or exponential $(1 - \alpha^c)$ regression. Figure 15 shows that the relative

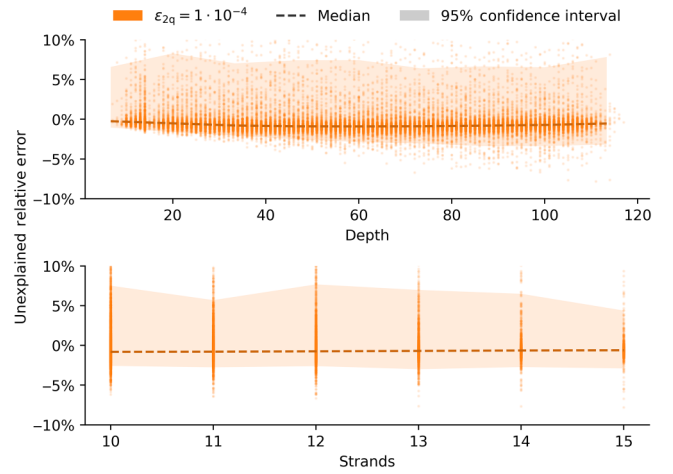


FIG. 15. The dependence of the relative error in Fig. 8 unexplained by the power-law fit for a set of 24k randomly generated benchmark braids, when simulated with a depolarizing noise model and $\epsilon_{2q} = 1 \times 10^{-4}$. While the unexplained relative error can be relatively large (5%–10%), the median is low ($<1\%$), and it is not correlated with either the depth of the circuits or the number of strands in each braid. This suggests that the total relative error of `cfev` depends mainly on the number of crossings.

error unexplained by the power-law fit does not depend strongly on the depth of the braid or the number of strands. Therefore, we did not believe it necessary to expand the benchmark set to include braids with more strands (which would have taken much more time to simulate).

2. Tensor-network methods

The benchmark set for `tn-proj`, `mpo-proj`, and `bracket` was comprised of 10k braids, generated as in Appendix D 1 with initial brick-wall depths between 5 and 50 in increments of 5 and between 10 and 30 strands. The braids were evenly distributed across all of these statistics. Since the time required to estimate the performance of `tn-proj`, `bracket`, and `mpo-proj` does not have a strong dependence on the number of strands (compared to the noisy simulations in the previous section), we elected to use a greater range of strands to have a more comprehensive benchmark.

We study the performance of `tn-proj` and `mpo-proj` by computing the weighted trace shown in Fig. 9. The description of the tensor network, including both the braid unitary and the projection matrices, is built up iteratively. The iteration is over the sequence of braid generators $U_{\sigma+}$ and $U_{\sigma-}$, here we do not use the compiled circuit representation of the braid generators, Fig. 4(b), instead we use the exact 3-qubit matrix, Eq. (1), where we set the action on the non-Fibonacci subspace to be a zero matrix. Once the braid is complete we similarly iterate over the projectors that apply the correct weightings in the trace. The inclusion of the projectors and the choice

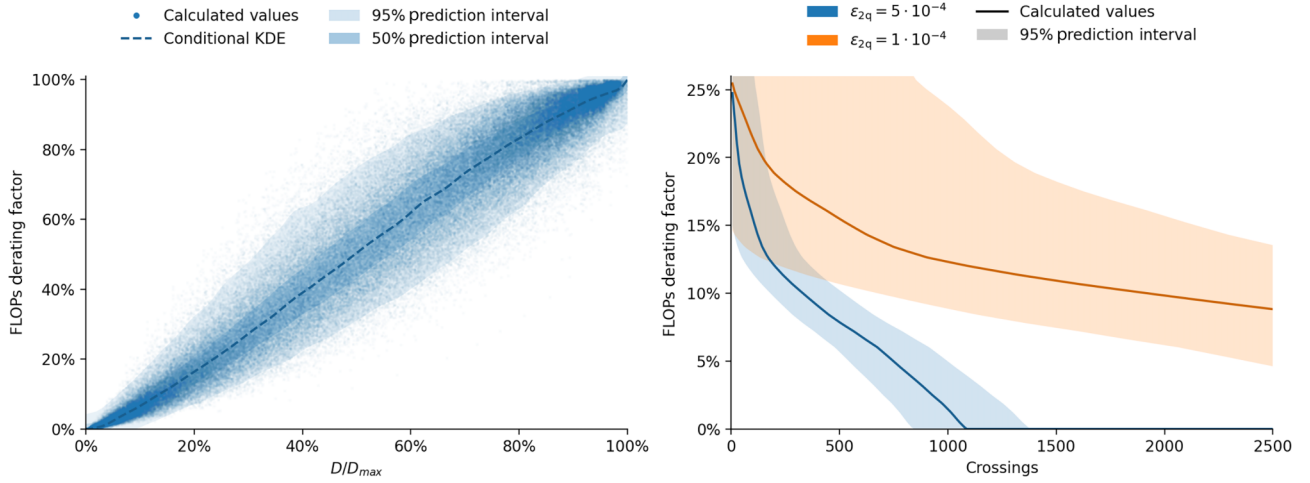


FIG. 16. *On the left:* The floating-point operations required to evaluate `mpo-proj` relative to the total number of operations when the bond dimension is maximum $\chi = \chi_{\max}$ so that no truncation occurs, as it depends on the level of truncation. The level of truncation is expressed as the amount of intermediate memory used relative to the $\chi = \chi_{\max}$ case. It was determined for a set of 10k braids by measuring the total floating point operations and intermediate memory usage for $\chi = 2^k$ with $k \in [3, 10]$. *On the right:* the amount of floating point operations needed for `mpo-proj` to achieve the same accuracy as `cfev`, as it depends on the number of crossings, relative to the number of floating point operations required for the $\chi = \chi_{\max}$ case. This is obtained by combining the left-hand plot with Fig. 10(a).

of the action of the braid generators on the non-Fibonacci subspace make the tensor-network nonunitary.

We can then evaluate this in two ways. For `tn-proj`, we use Cotengra [48] to find good contraction paths. We use the KaHyPar and random-greedy backends to Cotengra, configured to minimize total FLOPS. We used 128 hyper-optimization samples for each network. We also tested 1024 hyper-optimization samples on a subset of networks but did not observe any substantial improvement. We did not perform slicing of the network, but even if memory limitations are ignored, Fig. 11 indicates that this would not substantially change the boundaries for quantum advantage. We did not perform the actual contractions, but instead assumed that the calculated required FLOPS could be perfectly parallelized over our hardware targets. This is a conservative assumption that will bias the results in favor of classical algorithms. The results are shown in Fig. 20.

All MPO computations were carried out using the Quimb library [50]. At each step of the tensor network construction, we apply compression to reduce the maximum bond dimension to a fixed value χ . Once the compressed MPO is constructed in this way, we compute its trace and multiply by the prefactors to complete the estimate of the Jones polynomial for the Markov closure of the braid. From this, we can calculate the relative error by comparing against the known value of the Jones polynomial for the benchmark braids. We performed this for each braid for values of $\chi = 2^k$ with $k \in [3, 9]$. For each braid and for each k , we recorded the maximum intermediate size, and used this information to generate Fig. 10(a).

We found that $k = 13$ was sufficiently large that no compression needed to be applied for all braids in the benchmark set. The resources required in this case are given in Fig. 18. Thus for $k = 13$ for each braid, we recorded the dimensions of each bond after the application of each unitary to the MPO. The largest achieved bond dimension defines $\chi_{\max} \leq 2^{13}$ for that braid, and the largest total intermediate defines $D_{\max}$. Then using the theoretical FLOP counts listed in [51], we could calculate the theoretical number of FLOPS needed to evaluate the MPO for any intermediate χ value by replaying the sequence of operations performed by Quimb in the MPO computation, but using truncated bond dimensions to calculate the FLOP count for each operation. With this data, Figs. 8 and 10, we constructed Fig. 16, from which the time to solution of `mpo-proj` can be estimated. Again, we assumed that the calculated required FLOPS could be perfectly parallelized.

For `mpo-proj` we use a multilinear fit in Fig. 18 to extrapolate to larger numbers of crossings. Specifically, we take into account the number of crossings, strands, and depth of the braid. We fit to only the points with $c \geq 200$ because before this point the trend has not yet appeared to stabilize. We extrapolate along the principal axis of the dataset in terms of crossings, strands, and depth since this represents plausible larger instances generated in the same way as the original dataset.

For `tn-proj` in Fig. 20 (and `bracket` discussed in the next section), we use a multilinear fit that includes crossings c , strands $n - 1$, depth d , and also the term $\min(n, d)$. The reasoning behind this is that rectangular planar tensor networks can be contracted in time that scales

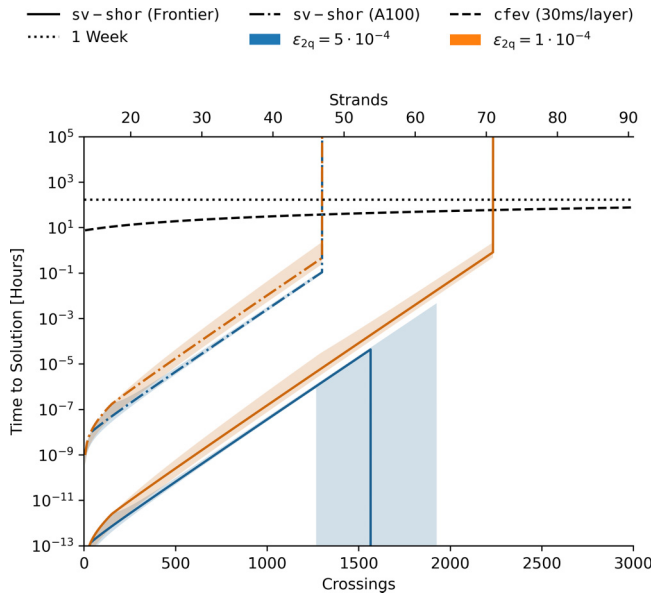


FIG. 17. Predicted time needed to calculate $V_{M(B)}(e^{i(2\pi/5)})$ using the `sv-shor` method, on the same benchmark set as in Fig. 10, to the same relative error achieved by `cf-ev`, as it depends on the number of crossings c and the 2-qubit error rate ϵ_{2q} . The solid lines represent computation on the Frontier cluster, while the dot-dashed lines represent the same computation on a single A100 GPU. The lines vertically downward represent the points where the relative error of `cf-ev` exceeds the 80% threshold given by Fig. 10, whereas the lines vertically upward represent the points where required memory exceeds the available memory of the classical hardware. For $\epsilon_{2q} = 5 \times 10^{-4}$, the relative error of `cf-ev` grows large enough that `sv-shor` is always faster than `cf-ev`. For $\epsilon_{2q} = 1 \times 10^{-4}$ the memory limit is reached before the point where `cf-ev` would become faster.

exponentially proportional to their shortest linear dimension (for instance, width or height). We would expect the networks generated by brick-wall style links to be roughly of this shape. More generally, planar tensor networks can be contracted in time that scales exponentially in the square root of the number of tensors [67]. This provides a more conservative scaling estimate than a multilinear fit of only c, n, d .

3. Classical knot-theoretic algorithms

There are various different classical algorithms for computing the Jones polynomial, or evaluating it at a certain point. Since doing this exactly is #P-hard [2], we would expect these methods to scale exponentially. Indeed, they can be divided broadly into two paradigms based on whether this scaling is in terms of strands or crossings.

The first group of algorithms perform computations with a 2^{n-1} dimensional vector (of Laurent polynomials or complex numbers, depending on the method). These can be derived for example through the connection to Hecke algebras [68] or quantum groups [69], and has been

implemented in practice [70]. Such methods are essentially identical to quantum algorithms for evaluating the Jones polynomial, and in particular `sv-shor`, as it performs a statevector simulation of each amplitude on the diagonal of U_B directly as a polynomial. Since they are not specialized to the $e^{i(2\pi/5)}$ evaluation point their cost scales as $O(2^n)$ in memory and $O(c4^n)$ in time.

The second group of algorithms expand the Jones polynomial as an exponential sum of terms using a skein relation (an equation that relates the original link to the links obtained by resolving a single crossing). For example, this includes direct expansion [71] (with complexity $O(2^c)$), methods based on “simple walks” [72,73], or the connection to the Tutte polynomial of graphs [74]. These computations can be optimized using dynamic programming, in a similar way to finding a good contraction path for a tensor network. Examples include [75] (with a stated complexity of at most $O(c2^{17\sqrt{c}})$), Bar-Natan’s algorithm for the Khovanov homology [76], and an algorithm due to Kauffman [1, p. 104-124] which we will call `bracket`. These methods are all abstractly similar to `tn-proj` in the same way that, taking another #P-hard problem as an example, model counting relates to tensor contraction [67]. Indeed, `tn-proj` scales exponentially in the width parameters (such as cut-width and tree-width) of the planar graph representing the quantum circuit [77], while [75,76] and `bracket` all scale exponentially in the width parameters of the planar graph representing the link (where each crossing is one vertex, and edges correspond to strands). This cannot be worse than the $O(2^n)$ scaling of the strand-based algorithms, so we choose to focus on these methods exclusively.

It is not a coincidence that both groups resemble different methods of simulating the corresponding quantum circuit; [15, Section 5.5] gives a heuristic argument that this may be the optimal classical algorithm for large knots, by showing that the Jones polynomial provides a general model of planar quantum circuits—the exact evaluation of the Jones polynomial corresponds to the exact simulation of postselected quantum circuits, in the same way that the APPROX-JONES-PLAT problem corresponds to the simulation of quantum circuits up to bounded error. Moreover, [38] shows that for random braids, U_B looks like a random quantum circuit. The theoretical scaling predicted for these algorithms is extremely poor, even making generous assumptions about the constant factors involved (e.g., for $c = 2000$ and $n = 80$, $c2^{17\sqrt{c}} \approx 10^{232}$, $c4^n \approx 10^{51}$), so this would not be a suitable point of comparison for `cf-ev`.

However, in the case of `bracket`, the method is given exactly by the contraction of a (nonunitary) tensor network over the ring of Laurent polynomials (that is, the elements of the tensors are Laurent polynomials). Thus, using Cotengra [48] to find good contraction paths for this network, and making the conservative assumption that each Laurent polynomial operation can be computed in

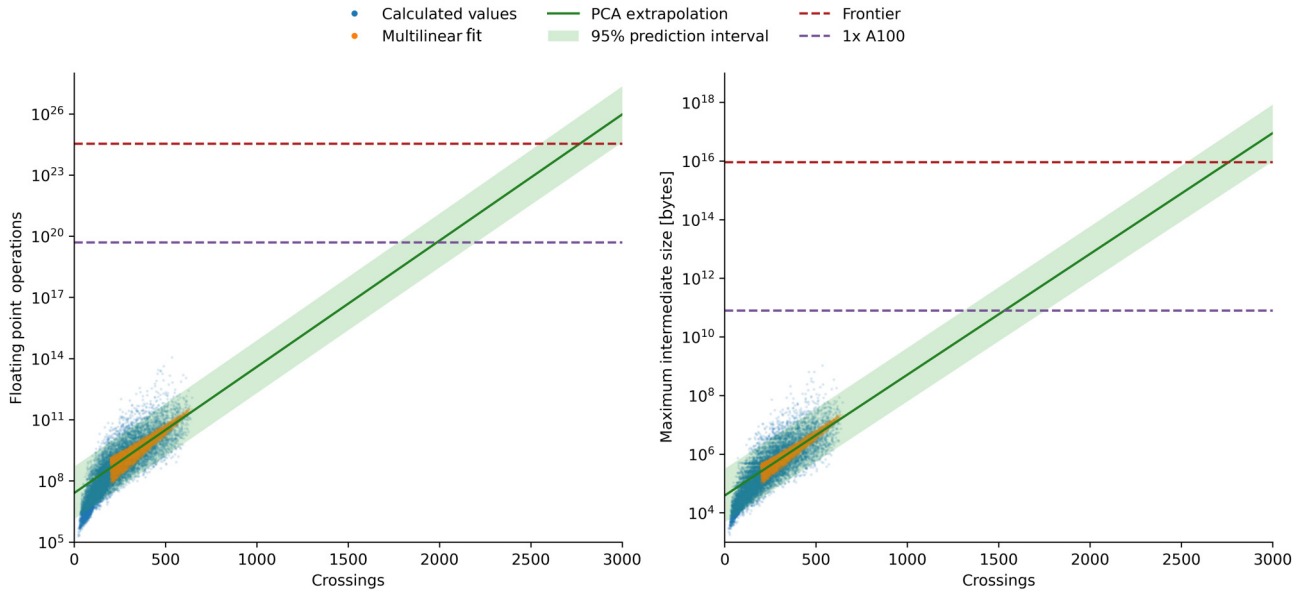


FIG. 18. The resources required to evaluate $V_{M(B)}(e^{i(2\pi/5)})$ using the `mpo-proj` method for the benchmark set as in Fig. 10. The bond dimension $\chi = \chi_{\max}$ was allowed to grow as large as needed to avoid any lossy compression. The extrapolation is a multilinear fit against the number of strands $n - 1$, crossings c , and depth of the braid for $c \geq 200$. It is extrapolated along the principal component of the dataset to represent possible performance for larger datasets generated in the same way. *On the left:* The number of floating point operations required. The dashed lines labeled “Frontier” and “1x A100” represent one month of computation on the respective hardware, for scale. *On the right:* The size of the largest intermediate tensor, in bytes. The dashed lines represent the memory limits of the respective hardware.

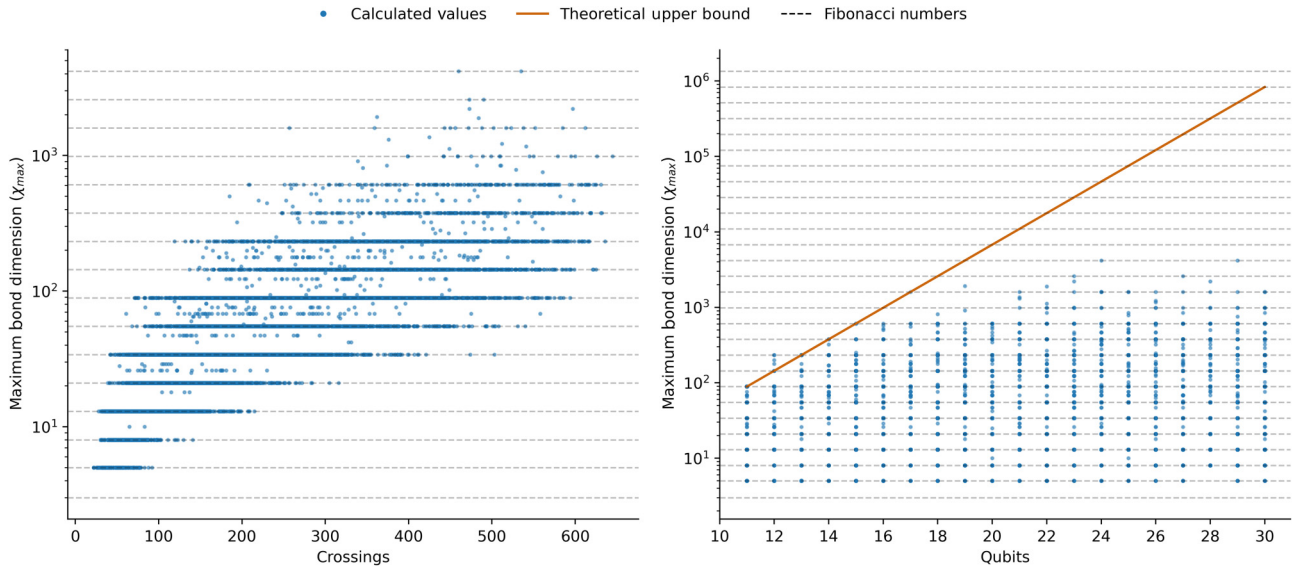


FIG. 19. The maximum bond dimension $\chi_{\max}$ required to evaluate $V_{M(B)}(e^{i(2\pi/5)})$ without any compression using the `mpo-proj` method for the benchmark set as in Fig. 10, as it depends on the number of qubits and the number of crossings. For almost all (96%) of instances, $\chi_{\max}$ is equal to a Fibonacci number. This suggests that `mpo-proj` is successful at reducing the bond dimension by exploiting that U_B acts nontrivially only on Fibonacci strings. Furthermore, $\chi_{\max}$ is usually much lower than the theoretical upper bound of f_n for n qubits, indicating that the complexity of braids in the benchmark set is likely limited by the number of crossings rather than the number of strands.

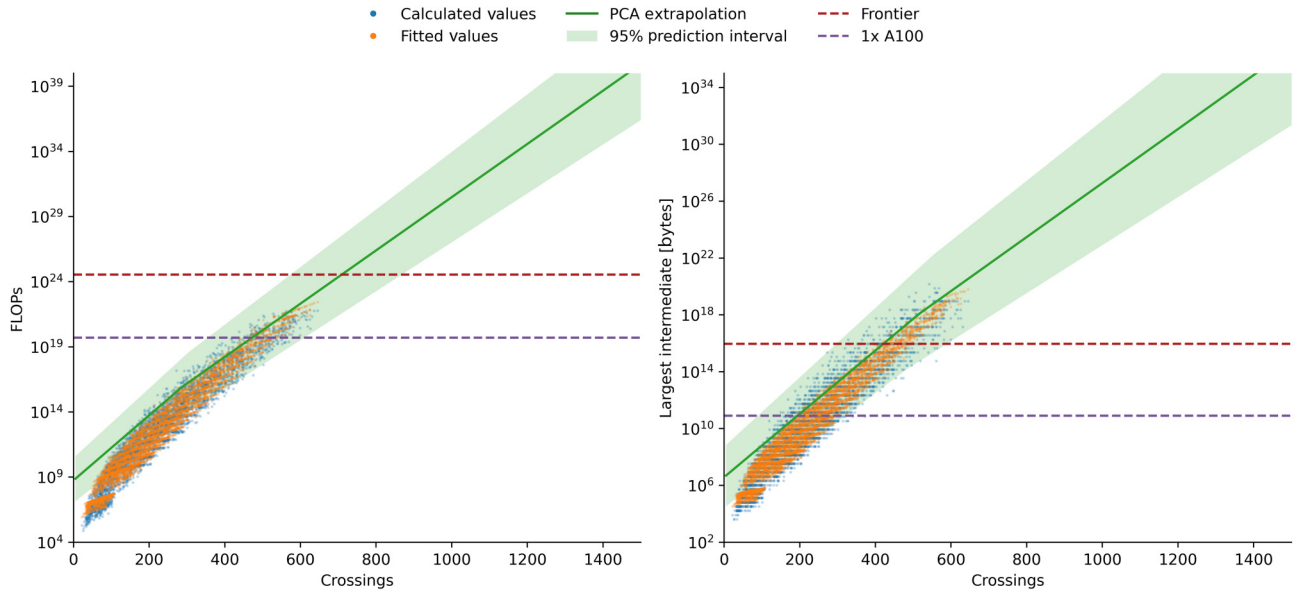


FIG. 20. The resources required to evaluate $V_{M(B)}(e^{i(2\pi/5)})$ using the `tn-proj` method for the benchmark set as in Fig. 10. The extrapolation is a multilinear fit against the number of strands $n - 1$, crossings c , depth of the braid d , and $\min(n, d)$ (see Appendix F 2 for more details). It is extrapolated along the principal component of the dataset to represent possible performance for larger datasets generated in the same way. The abrupt change in slope around $c \sim 500$ is due to the $\min(n, d)$ term and results in a more conservative estimate. *On the left:* The number of floating point operations required. The dashed lines labeled “Frontier” and “1x A100” represent one month of computation on the respective hardware, for scale. *On the right:* The size of the largest intermediate tensor, in bytes. The dashed lines represent the memory limits of the respective hardware.

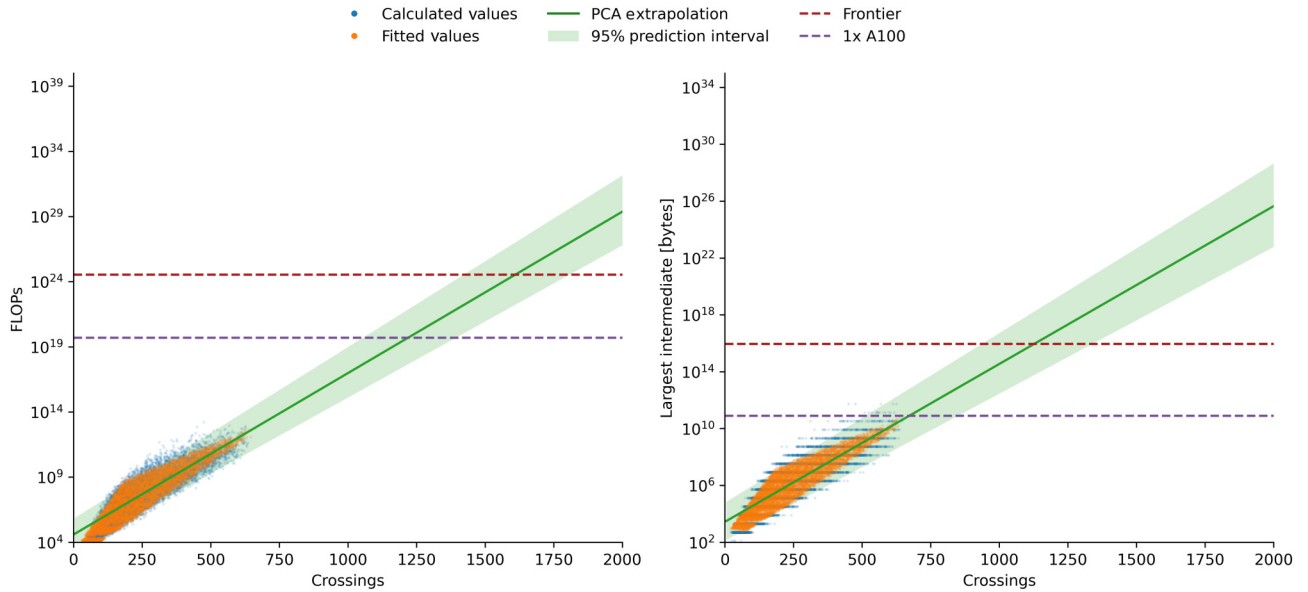


FIG. 21. The resources required to evaluate $V_{M(B)}(e^{i(2\pi/5)})$ using the `bracket` method for the benchmark set as in Fig. 10. This results from the optimization of the tensor-network contraction path, making the conservative estimate that each tensor element takes 8 bytes of memory and each corresponding arithmetic operation costs only one floating-point operation, see Appendix F 3 for more details. The extrapolation is a multilinear fit against the number of strands $n - 1$, crossings c , depth of the braid d , and $\min(n, d)$ (see Appendix F 2 for more details). It is extrapolated along the principal component of the dataset to represent possible performance for larger datasets generated in the same way. *On the left:* The number of floating point operations required. The dashed lines labeled “Frontier” and “1x A100” represent one month of computation on the respective hardware, for scale. *On the right:* The size of the largest intermediate tensor, in bytes. The dashed lines represent the memory limits of the respective hardware.

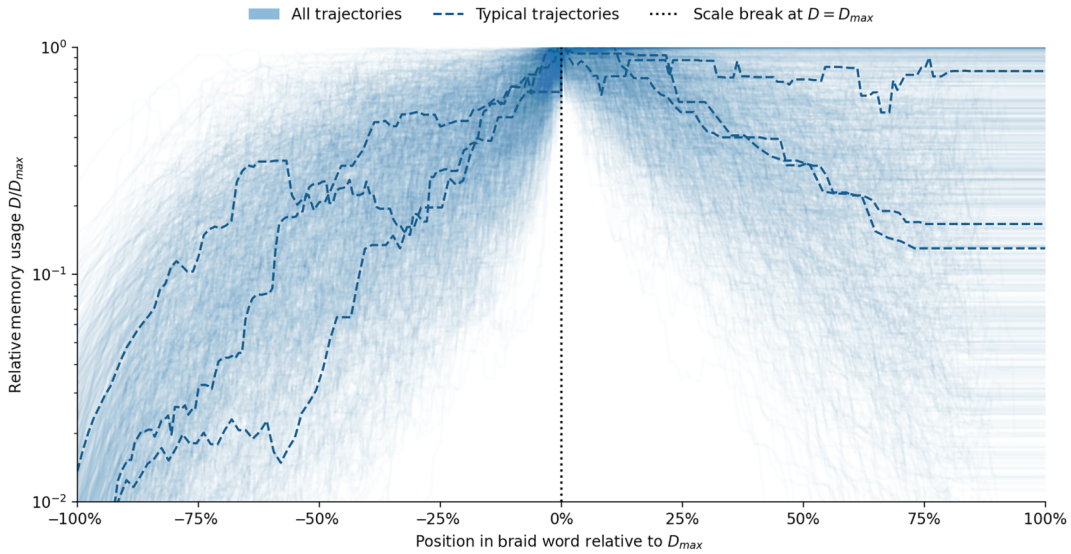


FIG. 22. The memory usage required to evaluate $V_{M(B)}(e^{i(2\pi/5)})$ without any compression using the `mpo-proj` method for each braid in the benchmark set (as in Fig. 10) as it evolves throughout the computation. We recorded the memory usage after each braid generator is applied, and normalize it relative to the maximum memory usage. The relative position in the braid word is measured in two different scales: -100% to 0% corresponds to all generators before the point $D = D_{\max}$ of maximum memory usage, and 0% to 100% is the same for all generators after that point. Three typical trajectories are shown with dashed lines. The memory usage increases roughly linearly (on a log-scale) up to the maximum before decreasing in the same way. This suggests that `mpo-proj` may be able to “see” the conjugate ABA^{-1} structure of the braids in the benchmark set (see Appendix D 1 for details), and hence it may perform differently for braids with a different structure.

only one floating-point operation, we can find a practical upper bound for the performance of `bracket`. We used the same contraction finding hyper-parameters as specified for `tn-proj` in the previous section. Since no large-scale implementations of any of these algorithms are available (e.g., [72] has $c \leq 12$, [70] has $n \leq 13$, [76] has $c \sim 50$), we will take this upper bound as representative of the best possible performance for all classical algorithms. Figure 21 shows the upper bound calculated in this way for the benchmark braid set described in the previous section.

4. Limitations

There are several limitations of these numerical studies which are worth mentioning. Most importantly, the results in Figs. 21, 8, and especially Fig. 10 require significant extrapolation outside the range of data we were able to compute in order to estimate where advantage might occur. In the absence of a strong mathematical reason why linear and power-law fits ought to hold in general for these quantities, we cannot say with certainty where the threshold for time to solution or energy-efficiency advantage for `cfev` over classical algorithms will occur, or even if it necessarily must occur for the ϵ_{2q} we are considering. The lack of further data in the case of `mpo-proj` is partially due to the fact that larger braids become difficult to benchmark on a single GPU due to Video Random Access Memory

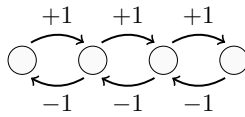
(VRAM) constraints. This is promising as it suggests that this problem is indeed difficult to solve classically, but also means we must be less certain about our conclusions. Secondly, in using benchmark braids constructed according to the method in Appendix D 1, we cannot necessarily draw strong conclusions about the behavior of braids that do not conform to this format (see Fig. 22). In the future, when a candidate braid for which advantage might be demonstrated is identified, it may be necessary to perform the same benchmarks with braids that more closely match this candidate.

APPENDIX G: GENERALIZATION TO OTHER ROOTS OF UNITY

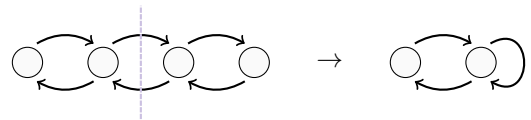
Our compilation of Shor and Jordan’s quantum algorithm [3] has the appealing property that the unitaries corresponding to each crossing are *local*, acting on only three qubits, and can thus be implemented in a hardware-efficient manner. On the other hand, the Aharonov-Jones-Landau (AJL) algorithm [4] can evaluate the Jones polynomial at any root of unity $e^{i(2\pi/k)}$, but the unitary for each crossing acts on *all* qubits (in the worst case), so it is unlikely to be realizable on NISQ machines. Therefore, it would be advantageous to generalize Shor and Jordan’s method so that roots of unity other than the fifth can be

evaluated while preserving locality of crossings. Here we sketch such a generalization, with the understanding that all the error mitigation, error detection, and compilation tricks described in this paper can likely be generalized to this case also. Our generalization is based on the connection between the two algorithms for the $k = 5$ case described in Appendix C of [3].

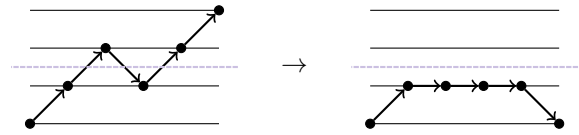
The AJL algorithm utilizes the *path representation* of the Temperley-Lieb algebra. In this model, given a braid of n strands evaluated at the $e^{i(2\pi/k)}$ root of unity with $k \geq 3$, the Hilbert space that is acted on by the crossings consists of all walks of length n on the path graph of $k - 1$ vertices that start at the left-most vertex. These paths are encoded as a list of transitions ± 1 , where $+1$ represents a step to the right, and -1 represents a step to the left. For example, in the $k = 5$ case, all the paths are generated as trajectories in the following finite state machine:



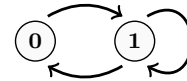
The inherent nonlocality in the AJL algorithm comes from the fact that the action of each crossing depends on position of the path at a given point, and determining this position requires inspecting all transitions up to that point. To obtain Shor and Jordan's representation, first note that the action of the crossings on each path (given by Equation 4 and Definition 2.5 in [4]) is symmetric about the center of the graph; that is, paths that land on vertex i or vertex $k - i - 2$ are treated exactly the same. Therefore, we can incorporate this symmetry into the finite state machine that generates these paths, by “folding” it over the line of symmetry,



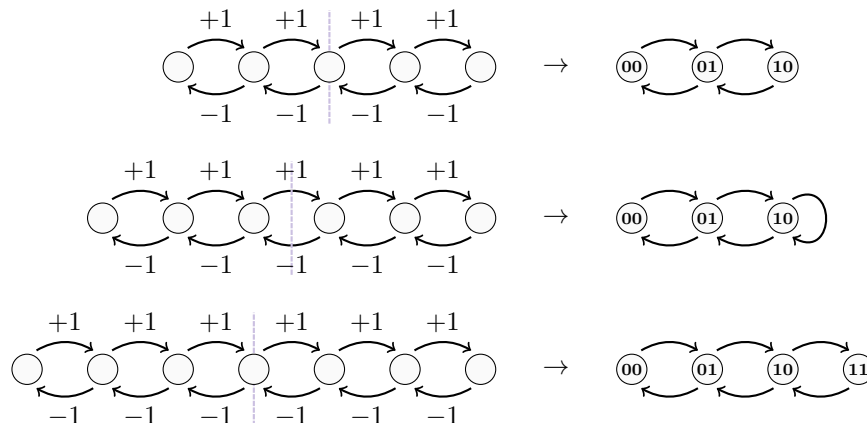
This can also be interpreted as “reflecting” the path over the line of symmetry (cf. Appendix C of [3]),



Then Shor and Jordan's representation is obtained by instead recording the *positions* of the path at each step, rather than the transitions between steps, and labeling each vertex in the state machine rather than each edge. This corresponds to the trajectories of the following finite state machine, which clearly generates only Fibonacci strings:



In the example above, this maps the path $\{+1, +1, -1, +1, +1\}$ to $\{0, 1, 1, 1, 0\}$. Recording the positions rather than the transitions enables crossings to be implemented by local gates, because the position at each step does not need to be calculated from the previous steps in the path. Therefore, to generalize this to $k > 5$, we can simply apply the same “folding” procedure to the corresponding path graph, and label each vertex. Note that for $k > 5$ the labels will require more than one qubit, since there are more than two vertices (a single qudit could also be used). For example for $k = 6, 7, 8$,



Then an n -strand braid will require $(n + 1)\lceil \log_2(\lfloor k/2 \rfloor) \rceil$ qubits to encode. Indeed this gives the $n + 1$ we see in Shor and Jordan's algorithm when $k = 5$, and also shows that $k = 2, 3, 4$ (but not $k = 6$) must be trivial to compute since the folded state machines only admit a single trajectory. This is asymptotically more qubits than the $n + O(\log_2(k))$ used in the AJL algorithm, but the savings due to the locality (in terms of gates) may be worth the trade-off.

Substituting these encodings into [4, Equation 4] then allows one to derive the equivalent of our Eq. (1) for $k > 5$. Concretely, consider all positions on the left half of the original path graph (by symmetry, this is sufficient), and all combinations of the following two ± 1 moves. This corresponds to a sequence of three positions in the reflected state machine, and the action of the Temperley-Lieb generators on this bit string can be read off from [4, Equation 4]. Substituting this into [4, Definition 2.5] then gives the action of a crossing on this bit string. For example for $k = 5$, we have

		Positions	
		1	2
Moves	$\{-1, -1\}$	...	...
	$\{-1, +1\}$	...	$U_\sigma 101\rangle = \phi^{-1} e^{i\frac{4\pi}{5}} 101\rangle + \phi^{-\frac{1}{2}} e^{-i\frac{3\pi}{5}} 111\rangle$
	$\{+1, -1\}$	$U_\sigma 010\rangle = e^{-i\frac{4\pi}{5}} 010\rangle$	$U_\sigma 111\rangle = -\phi^{-1} 111\rangle + \phi^{-\frac{1}{2}} e^{-i\frac{3\pi}{5}} 101\rangle$
	$\{+1, +1\}$	$U_\sigma 011\rangle = e^{i\frac{3\pi}{5}} 011\rangle$	$U_\sigma 110\rangle = e^{i\frac{3\pi}{5}} 110\rangle$

Which recovers Eq. (1) exactly. We also include the case of $k = 6$,

		Positions		
		1	2	3
Moves	$\{-1, -1\}$	...	...	$U_\sigma 100\ 100\rangle = e^{i\frac{19\pi}{12}} 100\ 100\rangle$
	$\{-1, +1\}$	...	$U_\sigma 010\ 001\rangle = \frac{e^{i\frac{\pi}{4}}}{\sqrt{3}} 010\ 001\rangle$ $+ \sqrt{\frac{2}{3}} e^{i\frac{5\pi}{12}} 011\ 001\rangle$	$U_\sigma 100\ 110\rangle = e^{i\frac{\pi}{4}} 100\ 110\rangle$
	$\{+1, -1\}$	$U_\sigma 000\ 100\rangle = e^{i\frac{\pi}{4}} 000\ 100\rangle$	$U_\sigma 011\ 001\rangle = \frac{e^{i\frac{\pi}{12}}}{\sqrt{3}} 011\ 001\rangle$ $+ \sqrt{\frac{2}{3}} e^{i\frac{5\pi}{12}} 010\ 001\rangle$	$U_\sigma 100\ 110\rangle = e^{i\frac{\pi}{4}} 100\ 110\rangle$
	$\{+1, +1\}$	$U_\sigma 000\ 110\rangle = e^{i\frac{19\pi}{12}} 000\ 110\rangle$	$U_\sigma 011\ 001\rangle = e^{i\frac{19\pi}{12}} 011\ 001\rangle$	$U_\sigma 100\ 100\rangle = e^{i\frac{19\pi}{12}} 100\ 100\rangle$

For higher k values, the corresponding matrix elements of the unitary representations can be derived similarly.

Critically, whenever $k \geq 4$, the string $00 \dots 0$ is not a valid trajectory of the state machine (in the same way that it is not a Fibonacci basis state), so it is not constrained by the above equations. Thus we can arrange for it to be an eigenstate of the crossing unitary, allowing `cfev` to be implemented essentially unchanged. Additionally, in the same way that we can efficiently sample from $\mathcal{F}_n$ classically to generate $|s\rangle$ bitstrings for `cfev`, AJL provides an algorithm [4, Claim 3.4] to sample paths uniformly for $k > 5$, which can then be “reflected” as above to be compatible with our representation.

-
- [1] Louis H. Kauffman, *Knots and Physics* (World Scientific, 2001), 3rd ed.
 - [2] F. Jaeger, D. L. Vertigan, and D. J. A. Welsh, On the computational complexity of the Jones and Tutte polynomials, *Math. Proc. Camb. Phil. Soc.* **108**, 35 (1990).
 - [3] Peter W. Shor and Stephen P. Jordan, Estimating Jones polynomials is a complete problem for one clean qubit, *Quantum Inf. Comput.* **8**, 681 (2008).
 - [4] Dorit Aharonov, Vaughan Jones, and Zeph Landau, in *STOC 2006* (2006).
 - [5] G. Passante, O. Moussa, C. A. Ryan, and R. Laflamme, Experimental approximation of the Jones polynomial with one quantum bit, *Phys. Rev. Lett.* **103**, 250501 (2009).

- [6] Louis H. Kauffman and Samuel J. Lomonaco Jr, The Fibonacci model and the Temperley-Lieb algebra, *Int. J. Mod. Phys. B* **22**, 5065 (2008).
- [7] Jiannis K. Pachos, *Introduction to Topological Quantum Computation* (Cambridge University Press, 2012), 1st ed.
- [8] Stefano Polla, Gian-Luca R. Anselmetti, and Thomas E. O'Brien, Optimizing the information extracted by a single qubit measurement, *Phys. Rev. A* **108**, 012403 (2023).
- [9] Chris N. Self, Marcello Benedetti, and David Amaro, Protecting expressive circuits with a quantum error detection code, *Nat. Phys.* **20**, 219 (2024).
- [10] Zhenyu Cai, Ryan Babbush, Simon C. Benjamin, Suguru Endo, William J. Huggins, Ying Li, Jarrod R. McClean, and Thomas E. O'Brien, Quantum error mitigation, *Rev. Mod. Phys.* **95**, 045005 (2023).
- [11] Chris N. Self, Sofyan Iblisdir, Gavin K. Brennen, and Konstantinos Meichanetzidis, Estimating the Jones polynomial for Ising anyons on noisy quantum computers, *Phys. Rev. Research* **8**, 023168 (2026).
- [12] Daniel Gottesman, and in *Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, edited by S. P. Corney, R. Delbourgo, P. D. Jarvis (International Press, Cambridge, MA, 1999), p. 32.
- [13] Alex Townsend-Teague and Konstantinos Meichanetzidis, Simplification strategies for the Qutrit ZX-calculus, [arXiv:2103.06914](https://arxiv.org/abs/2103.06914).
- [14] Oktay Göktas, W. K. Tham, Kent Bonsma-Fisher, and Aharon Brodutch, Benchmarking quantum processors with a single qubit, [arXiv:1905.05775](https://arxiv.org/abs/1905.05775).
- [15] Greg Kuperberg, How hard is it to approximate the Jones polynomial? *Theory Comput.* **11**, 183 (2015).
- [16] M. Bordewich, M. Freedman, L. Lovász, and D. Welsh, Approximate counting and quantum computation, *Comb. Probab. Comput.* **14**, 737 (2005).
- [17] Quantinuum, Quantinuum hardware specifications, 2025, <https://github.com/CQCL/quantinuum-hardware-specifications>
- [18] Kosuke Mitarai and Keisuke Fujii, Methodology for replacing indirect measurements with direct measurements, *Phys. Rev. Res.* **1**, 013006 (2019).
- [19] Adriano Barenco, André Berthiaume, David Deutsch, Artur Ekert, Richard Jozsa, and Chiara Macchiavello, Stabilisation of quantum computations by symmetrisation, [arXiv:quant-ph/9604028](https://arxiv.org/abs/quant-ph/9604028).
- [20] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf, Quantum fingerprinting, *Phys. Rev. Lett.* **87**, 167902 (2001).
- [21] Sitan Chen, Jordan Cotler, Hsin-Yuan Huang, and Jerry Li, The complexity of NISQ, *Nat. Commun.* **14**, 6001 (2023).
- [22] Tomoyuki Morimae, Keisuke Fujii, and Harumichi Nishimura, Power of one nonclean qubit, *Phys. Rev. A* **95**, 042336 (2017).
- [23] Peter W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* **26**, 1484 (1997).
- [24] Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, and Jay M. Gambetta, Validating quantum computers using randomized model circuits, *Phys. Rev. A* **100**, 032328 (2019).
- [25] Karl Mayer, Alex Hall, Thomas Gatterman, Si Khadir Halit, Kenny Lee, Justin Bohnet, Dan Gresh, Aaron Hankin, Kevin Gilmore, Justin Gerber, and John Gaebler, Theory of mirror benchmarking and demonstration on a quantum computer, [arXiv:2108.10431](https://arxiv.org/abs/2108.10431).
- [26] Jordan Hines, Marie Lu, Ravi K. Naik, Akel Hashim, Jean-Loup Ville, Brad Mitchell, John Mark Kriekebaum, David I. Santiago, Stefan Seritan, Erik Nielsen, Robin Blume-Kohout, Kevin Young, Irfan Siddiqi, Birgitta Whaley, and Timothy Proctor, Demonstrating scalable randomized benchmarking of universal gate sets, *Phys. Rev. X* **13**, 041030 (2023).
- [27] Thomas Lubinski, Sonika Johri, Paul Varosy, Jeremiah Coleman, Luning Zhao, Jason Necaise, Charles H. Baldwin, Karl Mayer, and Timothy Proctor, Application-oriented performance benchmarks for quantum computing, *IEEE Trans. Quantum Eng.* **4**, 1 (2023).
- [28] Daniel Mills, Seyon Sivarajah, Travis L. Scholten, and Ross Duncan, Application-motivated, holistic benchmarking of a full quantum computing stack, *Quantum* **5**, 415 (2021).
- [29] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson, Scalable and robust randomized benchmarking of quantum processes, *Phys. Rev. Lett.* **106**, 180504 (2011).
- [30] Scott Aaronson and Daniel Gottesman, Improved simulation of stabilizer circuits, *Phys. Rev. A* **70**, 052328 (2004).
- [31] G. C. Wick, The evaluation of the collision matrix, *Phys. Rev.* **80**, 268 (1950).
- [32] Richard Jozsa and Akimasa Miyake, Matchgates and classical simulation of quantum circuits, *Proc. R. Soc. A: Math. Phys. Eng. Sci.* **464**, 3089 (2008).
- [33] Robin Blume-Kohout and Kevin C. Young, A volumetric framework for quantum computer benchmarks, *Quantum* **4**, 362 (2020).
- [34] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz, A generative modeling approach for benchmarking and training shallow quantum circuits, *npj Quantum Inf.* **5**, 45 (2019).
- [35] Alexander J. McCaskey, Zachary P. Parks, Jacek Jakowski, Shirley V. Moore, Titus D. Morris, Travis S. Humble, and Raphael C. Pooser, Quantum chemistry as a benchmark for near-term quantum computers, *npj Quantum Inf.* **5**, 99 (2019).
- [36] Aliza U. Siddiqui, Kaitlin Gili, and Chris Ballance, Stressing out modern quantum hardware: Performance evaluation and execution insights, [arXiv:2401.13793](https://arxiv.org/abs/2401.13793).
- [37] Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout, Measuring the capabilities of quantum computers, *Nat. Phys.* **18**, 75 (2022).
- [38] Ryan L. Mann and Michael J. Bremner, On the complexity of random quantum computations and the Jones polynomial, [arXiv:1711.00686](https://arxiv.org/abs/1711.00686).
- [39] Lorenza Viola, Emanuel Knill, and Seth Lloyd, Dynamical decoupling of open quantum systems, *Phys. Rev. Lett.* **82**, 2417 (1999).
- [40] Peter J. Diggle and Peter Hall, A fourier approach to non-parametric deconvolution of a density estimate, *J. R. Stat. Soc.: Ser. B* **55**, 523 (1993).
- [41] E. A. Nadaraya, On estimating regression, *Theory Probab. Appl.* **9**, 141 (1964).

- [42] Raphael A. Meyer, Cameron Musco, Christopher Musco, and David P. Woodruff, in *Symposium on Simplicity in Algorithms (SOSA)* (2021), p. 142.
- [43] Top500 Authors, *JEDI*, <https://top500.org/system/180269/> [Accessed 23-01-2025] (2024).
- [44] Matthew DeCross *et al.*, The computational power of random quantum circuits in arbitrary geometries, *Phys. Rev. X* **15**, 021052 (2025).
- [45] Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky, NVIDIA A100 Tensor Core GPU: Performance and innovation, *IEEE Micro* **41**, 29 (2021).
- [46] Scott Atchley *et al.*, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2023), p. 1.
- [47] S. A. Moses *et al.*, A race-track trapped-ion quantum processor, *Phys. Rev. X* **13**, 041052 (2023).
- [48] Johnnie Gray and Stefanos Kourtis, Hyper-optimized tensor network contraction, *Quantum* **5**, 410 (2021).
- [49] Satoshi Imamura, Masafumi Yamazaki, Takumi Honda, Akihiko Kasagi, Akihiro Tabuchi, Hiroshi Nakao, Naoto Fukumoto, and Kohta Nakashima, mpiQulacs: A distributed quantum computer simulator for A64FX-based cluster systems, [arXiv:2203.16044](https://arxiv.org/abs/2203.16044).
- [50] Johnnie Gray, Quimb: A python library for quantum information and many-body calculations, *J. Open Source Softw.* **3**, 819 (2018).
- [51] Susan Blackford, LAPACK benchmark, <https://www.netlib.org/lapack/lug/node71.html> [Accessed 15-01-2025] (1999).
- [52] Tuomas Laakkonen, Chris N. Self, and Enrico Rinaldi, Jones-polynomial, <https://github.com/CQCL/jones-polynomial> (2025).
- [53] Adam Zsolt Wagner, Constructions in combinatorics via neural networks, [arXiv:2104.14516](https://arxiv.org/abs/2104.14516).
- [54] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis, and Pushmeet Kohli, Advancing mathematics by guiding human intuition with AI, *Nature* **600**, 70 (2021).
- [55] P. D. Bangert, M. A. Berger, and R. Prandi, In search of minimal random braid configurations, *J. Phys. A: Math. Gen.* **35**, 43 (2002).
- [56] Sergei Gukov, James Halverson, Fabian Ruehle, and Piotr Sułkowski, Learning to unknot, *Mach. Learn.: Sci. Technol.* **2**, 025035 (2021).
- [57] Louis H. Kauffman and Samuel J. Lomonaco Jr., in *Quantum Information and Computation VIII* (2010), Vol. 7702, p. 770203.
- [58] Alexander Schmidhuber, Michele Reilly, Paolo Zanardi, Seth Lloyd, and Aaron Lauda, A quantum algorithm for Khovanov homology, [arXiv:2501.12378](https://arxiv.org/abs/2501.12378).
- [59] Iosif Pinelis, Concentration inequality for square roots. MathOverflow, <https://mathoverflow.net/q/455546> (version: 2023-09-29) (2023).
- [60] Pierre Vogel, Representation of links by braids: A new algorithm, *Comment. Math. Helv.* **65**, 104 (1990).
- [61] Sang Youl Lee and Myoungsoo Seo, A formula for the braid index of links, *Topol. Appl.* **157**, 247 (2010).
- [62] M. S. Paterson and A. A. Razborov, The set of minimal braids is co-NP-complete, *J. Algorithms* **12**, 393 (1991).
- [63] Thomas A. Gittings, Minimum braids: A complete invariant of knots and links, [arXiv:math/0401051](https://arxiv.org/abs/math/0401051).
- [64] M. A. Berger, Minimum crossing numbers for 3-braids, *J. Phys. A: Math. Gen.* **27**, 6205 (1994).
- [65] Philippe Tillet, Hsiang-Tsung Kung, and David Cox, in *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages* (2019), p. 10.
- [66] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta, Quantum computing with Qiskit, [arXiv:2405.08810](https://arxiv.org/abs/2405.08810).
- [67] Stefanos Kourtis, Claudio Chamon, Eduardo R. Mucciolo, and Andrei E. Ruckenstein, Fast counting with tensor networks, *SciPost Phys.* **7**, 060 (2019).
- [68] H. R. Morton and H. B. Short, Calculating the 2-variable polynomial for knots presented as closed braids, *J. Algorithms* **11**, 117 (1990).
- [69] Peter Tingley, A minus sign that used to annoy me but now I know why it is there, [arXiv:1002.0555](https://arxiv.org/abs/1002.0555).
- [70] H. R. Morton, Liverpool university knot theory group programs and procedures, <https://www.liverpool.ac.uk/sul4/knotprogs.html> [Accessed 06-03-2025] (2014).
- [71] Louis H. Kauffman, State models and the Jones polynomial, *Topology* **26**, 395 (1987).
- [72] Mustafa Hajij and Jesse Levitt, An efficient algorithm to compute the colored Jones polynomial, [arXiv:1804.07910](https://arxiv.org/abs/1804.07910).
- [73] Hans U. Boden and Matthew Shimoda, Braid representatives minimizing the number of simple walks, *Ars Math. Contemp.* **23**, 1 (2023).
- [74] Kyoko Sekine, Hiroshi Imai, and Seiichiro Tani, in *Proceedings of the 6th International Symposium on Algorithms and Computation* (1995), p. 224.
- [75] Lauren Ellenberg, Gabriella Newman, Stephen Sawin, and Jonathan Shi, Efficient computation of the Kauffman bracket, [arXiv:1303.7179](https://arxiv.org/abs/1303.7179).
- [76] Dror Bar-Natan, Fast Khovanov homology computations, [arXiv:math/0606318](https://arxiv.org/abs/math/0606318).
- [77] Bryan O’Gorman, in *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)* (2019), Vol. 135, p. 10:1.