Transversal injection for direct encoding of ancilla states for non-Clifford gates using stabilizer codes

Jason Gavriel,[1,2,*] Daniel Herr,[3] Alexis Shaw,[1,2] Michael J. Bremner,[1,2] Alexandru Paler,[4] and Simon J. Devitt[2]
[1]*Centre for Quantum Computation and Communication Technology, New South Wales 2052, Australia*
[2]*Centre for Quantum Software and Information, University of Technology Sydney, Sydney, New South Wales 2007, Australia*
[3]*d-fine GmbH, An der Hauptwache 7, 60213 Frankfurt, Germany*
[4]*Department of Computer Science, Aalto University, 02150 Espoo, Finland*

(Received 7 December 2022; revised 14 December 2022; accepted 12 May 2023; published 10 July 2023)

Fault-tolerant, error-corrected quantum computation is commonly acknowledged to be crucial to the realization of large-scale quantum algorithms that could lead to extremely impactful scientific or commercial results. Achieving a universal set of quantum gate operations in a fault-tolerant, error-corrected framework suffers from a conservation of unpleasantness. In general, no matter what error-correction technique is employed, there is always one element of a universal gate set that carries a significant resource overhead—either in physical qubits, computational time, or both. Specifically, this is due to the application of non-Clifford gates. A common method for realizing these gates for stabilizer codes such as the surface code is a combination of three protocols: state injection, distillation, and gate teleportation. These protocols contribute to the resource overhead compared with logical operations such as a CNOT gate and contribute to the qubit resources for any error-corrected quantum algorithm. In this paper, we introduce a very simple protocol to potentially reduce this overhead for non-Clifford gates: transversal injection. Transversal injection modifies the initial physical states of all data qubits in a stabilizer code before standard encoding and results in the direct preparation of a large class of single qubit states, including resource states for non-Clifford logic gates. Preliminary results hint at high-quality fidelities at larger distances and motivate further research on this technique.

DOI: 10.1103/PhysRevResearch.5.033019

I. INTRODUCTION

Quantum error correction (QEC) forms a crucial component of large-scale quantum computing systems [1,2]. The difficulty in fabricating ultralow-error-rate qubits and quantum gates at scale necessitates active techniques to mitigate errors caused by environmental decoherence, fabrication errors, measurement and control errors, and components that are always probabilistic [3]. While there is currently a focus on the so-called noisy intermediate-scale quantum (NISQ) regime [4]—where it is hoped that a scientifically or commercially valuable quantum algorithm can be found that is small enough not to require QEC on the current or next-generation quantum computing chipsets—most theoretical work suggests that the true value in quantum computing will lie with large simulation algorithms that will unarguably require extensive error correction [5–7], unless we see a significant revolution in hardware technology.

While work on QEC is extensive [8], the physical constraints on quantum hardware architectures has resulted in the dominance of one type of QEC code, namely, the surface code [9]. Defined over a two-dimensional (2D) nearest-neighbor array of physical qubits, it is now the most studied QEC code and the preferred model for numerous architecture blueprints in multiple hardware platforms [10–14].

However, the implementation of QEC for any quantum algorithm, large or small, comes with a significant overhead in physical qubits and/or computational time [15]. This is not surprising, as the goal of QEC is often to take a physical error rate of the hardware of $p = 10^{-3} \rightarrow 10^{-4}$ and reduce it by many orders of magnitude, with large-scale quantum simulation estimated to require error rates of 10^{-20} or even lower [6,16].

Authors of theoretical work in QEC, algorithmic design, compilation, and resource optimization have done a surprising job of figuring out better and better ways to implement error-corrected algorithms [17–19], with one of the most studied algorithms, Shor's algorithm, being a useful example. Early compilation efforts, with the surface code, benchmarked Shor's algorithm at the beginning of the 2010s, showing that upwards of 30 billion components would be required to implement Shor-2048 [20]. By focusing entirely on better ways to implement both the algorithm itself and the underlying QEC protocols, this has been reduced to 20 million qubits by the end of the 2010s without changing any assumptions at the physical hardware level [15].

How much this can still be reduced depends on several factors, even when we still do not change the hardware assumptions of the underlying microarchitecture. The first is just the raw qubit overhead to encode a logical qubit of information up to some desired logical error rate. For a distance

*jason@gavriel.au

d error-correction code, the logical error rate scales as $p_L \approx O(p^{\lfloor (d-1)/2 \rfloor})$, under a simple symmetric Pauli model. This assumes that the physical error rate of all parts of the hardware system (decoherence, control, measurement, etc.) is under the fault-tolerant threshold of the code, $p \approx 0.67\%$ for the surface code [21]. If we take a square, unrotated, planar surface code, the total number of qubits (data + syndrome qubits) scales as $N = (2d-1)^2$, hence $p_L \approx O(p^{\lfloor (\sqrt{N}-1)/4 \rfloor})$. This exponential scaling means that, for a heavily error-corrected code, a lattice of $N > 1000$ is required [22]. How much this base-level logical qubit overhead can be reduced, while still having a code that is architecturally feasible, is still an open question.

In this paper, we introduce a simple way to produce encoded non-Pauli eigenstates. This process we dub *transversal injection* modifies the way in which non-Clifford ancillary states are encoded. A standard approach for creating logical qubits is to initialize data qubits into the $|0\rangle$ or $|+\rangle$ state and measure the stabilizers of the code [9]. If all stabilizers commute or anticommute in the desired fashion, we have successfully encoded a logical qubit in the $|0\rangle$ or $|+\rangle$ state, respectively. Transversal injection involves performing a transversal rotation—a uniform single-qubit rotation on all data qubits individually—initializing them in some non-Pauli state. This is followed on by the standard stabilizer measurement procedure. During the encoding state, the stabilizers will either commute or anticommute, and the encoded logical state will then be some non-Pauli eigenstate. The string of all stabilizer measurements forms what we will call a *stabilizer trajectory*, which can be used to determine the resultant state.

In this paper, we perform simulations of this protocol under the influence of physical errors and investigate how the encoded error rates are related under a standard Pauli noise model on the circuit level. A modest postselection strategy is also applied to improve the fidelity of this protocol. We show through these preliminary results that we can generate states with a higher fidelity than $1-p$, where p is our physical error rate. This potentially eases the resource overhead of state distillation. Further research is needed to investigate fidelities at higher distances codes, further optimization of the classical algorithm, and compilation strategies.

We present this technique in the context of the surface code, but it should be stressed that it is applicable to all stabilizer-based QEC codes. The contents of this paper include

(1) A description of the formalism.

(2) A classical algorithm for calculating the logically encoded state.

(3) Numerical simulations of transversal injection on the surface code.

(4) Discussion of this technique and implications for QEC circuit compilation.

This method can be looked at as the qubit extension of what was found in the continuous variable context [23], where all-Gaussian universality was discovered in the context of the Gottesman-Kitaev-Preskill code.

II. STATE DISTILLATION

Arguably the largest source of overhead in fault tolerance is ancillary protocols required to perform error-corrected logic operations such as lattice surgery [24] and complex compound protocols to create resource states for universal computation [17,25,26]. Some gates have a transversal realization in the code which is inherently fault tolerant and require few if any additional resources to implement. Eastin and Knill [27] proved that there is no QEC code that can perform a universal set of transversal gates while correcting an arbitrary error. The no-go of Eastin and Knill [27] can be worked around when not restricted to using a single code, such as gauge-fixed code conversion [28]. A more common approach is simply to hack together a fault-tolerant implementation of the nontransversal gate via a sequence of state injection, magic state distillation, and gate teleportation [25,29]. One example of state distillation is the quantum Reed-Muller code which takes 15 magic states with a respective error rate of p and distils them into a single state with an error rate of $35p^3$. This process can be performed recursively to reach a desired level of fidelity.

The encoding of a magic state for use in state distillation has a p that is dependent on the fidelity of the single-qubit state and the two-qubit gates that are needed to realize the encoding. Errors propagate in this protocol and cause the logically encoded state to have approximately the same logical error rate as the physical qubit and gates used for the injection, irrespective of the amount of error correction used for the encoding [30]. Even small improvements at this stage of the protocol will be compounded by multiple state distillation steps, achieving a significant improvement in fidelity or a reduction in the amount of distillation needed for a target fidelity level. Any gain in fidelity seen through transversal injection could outweigh any complexity introduced by its probabilistic nature elsewhere in algorithm compilation.

III. PAULI EIGENSTATE ENCODING

The procedure of transversal injection is only a minor alteration to standard surface code operation and is derived from a simple observation about how Pauli eigenstates are encoded. This section will briefly cover how a single high-fidelity logical state can be encoded using multiple physical qubits. This is a requisite for understanding how transversal injection works, as described in the next section.

Traditionally, we only consider two logically encoded states in the surface code that can be initialized, the $|0\rangle_L$ and $|+\rangle_L$ states. These two states are eigenstates of the logical Z_L and X_L operators of the planar surface code and are hence natural to consider when examining encoded state initialization. The encoding procedure for each of these two states proceeds in a similar manner; we will use the $|0\rangle_L$ state for ease of discussion.

When a logical qubit in the planar surface code is initialized into $|0\rangle_L$, we first initialize each of the physical qubits in the data block in the $|0\rangle$ state. Hence, our system can be described in terms of the following stabilizer matrix:

$$\begin{bmatrix} Z_1 & I_2 & I_3 & \dots & I_N \\ I_1 & Z_2 & I_3 & \dots & I_N \\ I_1 & I_2 & Z_3 & \dots & I_N \\ . & . & . & \dots & . \\ I_1 & I_2 & I_3 & \dots & Z_N \end{bmatrix}, \tag{1}$$

where N is the number of data qubits in the code, and the eigenvalue of each stabilizer is 0.

Initialization of the encoded state then requires repeated measurements of each of the X-type vertex stabilizers of the surface code [9]. Each of these stabilizer measurements results in a random parity, and the new stabilizer set will contain these X-type vertex stabilizers and combinations of the stabilizers in Eq. (1) that commute with all the X-type stabilizers that were just measured.

By definition, there are only two sets of Z operators that commute with all the X-type stabilizers, namely, all the Z-type plaquette stabilizers of the code and the Z-chain operator that defines the Z eigenstate of the encoded qubit. The parity of these commuting operators is defined by the parity of the individual stabilizers of the physical $|0\rangle$ states that are the rows of Eq. (1).

Hence, after the X-type vertex stabilizers are measured (and their resultant syndromes decoded to perform the required Z correction to transform the eigenvalues of the X stabilizers into all zeros), the stabilizer matrix will be

$$\begin{bmatrix} X_{v_i} \\ . \\ Z_{p_i} \\ Z_L \end{bmatrix}, \tag{2}$$

where $\{X_{v_i}, Z_{p_i}\}$ for $i \in [1, (N-1)/2]$ are the stabilizers of the surface code, and Z_L is the chain operator that runs across the lattice, defining the logical state.

The fact that each physical qubit starts in a 0 eigenstate of the Z operator means that, not only is it unnecessary to measure the Z-type plaquette stabilizers when initializing an encoded qubit, but the resulting logical state will automatically be in a 0 eigenstate of Z_L. While in principle the Z stabilizers do not need to be measured, in practice, they are, as once the qubit is initialized, it needs to be corrected against possible physical X errors.

As the parities of the resultant Z stabilizers in Eq. (2) are determined by the product of the individual parities of the Z stabilizers in Eq. (1), you can derive what you would need to initialize a $|1\rangle_L$ state directly. In the case of the surface code, initializing in the $|1\rangle_L$ state requires initializing a subset of the physical qubits (e.g., along a diagonal of the lattice) in the $|1\rangle$ state, not all the physical qubits (which would actually initialize the logical qubit back into the $|0\rangle_L$ state).

IV. TRANSVERSAL INJECTION

Transversal injection is based on the realization that the traditional standard procedure is only a small subset of what is actually possible. For example, when starting with all physical qubits in the $|0\rangle$ state, encoding a $|0\rangle_L$ state can be done reliably, as we are already in an eigenstate of the Z portion of our stabilizer group. If instead our data qubits are all initialized in some arbitrary state $|\chi\rangle$, we do not simply encode our logical qubit into the same state as our physical states. The stabilizer group will then commute or anticommute in a probabilistic fashion, and the measured eigenvalues will allow us to infer what our encoded logical state is.

A. Initialization

Let us consider the case where all the data qubits are initially placed into the states:

$$|\chi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{3}$$

i.e., we first perform a transversal rotation on all data qubits to the state $|\chi\rangle$ before we follow the same procedure to encode a $|0\rangle_L$ or $|+\rangle_L$ state.

When considering the system of N data qubits that have been uniformly transformed, the values of each eigenstate are naturally related to the Hamming weight of its integer value and the chosen transversal operation:

$$|\chi\rangle^{\otimes N} = \sum_{n=0}^{2^N-1} \alpha^{N-\hat{H}(n)} \beta^{\hat{H}(n)} |n\rangle\,, \tag{4}$$

where n is the integer representation of the eigenvalues of the system, and $\hat{H}(n)$ is the Hamming weight or number of bits set to 1 in its binary representation.

B. Syndrome extraction

The set of measurements for each stabilizer $\{X_{v_i}, Z_{p_i}\}$ defines a stabilizer trajectory. This set is denoted by the string $\{X_1, \ldots, X_{(N-1)/2}, Z_1, \ldots, Z_{(N-1)/2}\} \in \{0, 1\}$ of eigenvalues that are measured for the $N-1$ X-type vertex and Z-type plaquette stabilizers. This definition will be used throughout this paper to reference a set of measurement outcomes that together determine a resultant logical state. In the absence of physical errors, the X projections are probabilistic, and the Z projections are deterministic when encoding $|0\rangle_L$ or vice versa for the $|+\rangle_L$ state. In transversal injection the outcomes of all measurements are probabilistic and will select out only certain eigenvalues that are consistent with the previously observed measurements.

The first step in encoding is to measure all the X-type stabilizers, X_{v_i}. Our initial state is projected into eigenstates of the X stabilizers with the eigenvalue for each operator determined by the qubit measurements of each stabilizer circuit. The measurement of these stabilizers will result in an eigenvalue sequence for the X-type stabilizers and cause perturbations in the superposition of the Z-type stabilizers until they too are measured. The second step is to project into eigenstates of the Z stabilizers in the same manner.

The stabilizer tableau is defined below with the first $(N-1)/2$ rows referring to the X-type and the second $(N-1)/2$ rows referring to the Z-type stabilizer sets [31]. The first N columns are set to 1 to indicate an X Pauli operator on each respective qubit. A 1 in the second N columns indicates the presence of a Z operator on each respective qubit. By

definition for the planar surface code, the X-type stabilizers only contain X Pauli elements and vice versa for the Z sector. The last row is the chain operator, spanning the lattice, that denotes the logical Z operator:

$$\left[\begin{array}{ccc|ccc|c} x_{1,1} & \cdots & x_{1,N} & z_{1,1} & \cdots & z_{1,N} & X_{v_1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{(N-1)/2,1} & \cdots & x_{(N-1)/2,N} & z_{(N-1)/2,1} & \cdots & z_{(N-1)/2,N} & X_{v_{(N-1)/2}} \\ \hline x_{(N+1)/2,1} & \cdots & x_{(N+1)/2,N} & z_{(N+1)/2,1} & \cdots & z_{(N+1)/2,N} & Z_{p_1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N-1,1} & \cdots & x_{N-1,N} & z_{N-1,1} & \cdots & z_{N-1,N} & Z_{p_{(N-1)/2}} \\ \hline x_{L,1} & \cdots & x_{L,N} & z_{L,1} & \cdots & z_{L,N} & Z_L \end{array}\right]. \tag{5}$$

As an example, consider a distance $d = 2$ unrotated code (Fig. 1), with stabilizers $XXXII$, $IIXXX$, $ZIZZI$, and $IZZIZ$ and logical chain $ZZIII$. The tableau is populated by a 0 or 1, where a 1 indicates a Pauli X or Z is applied on that qubit. Below is an example of a tableau for the trivial case of initializing in the $|0\rangle_L$ state where each stabilizer has an eigenvalue of 0 as indicated by the final column:

$$\left[\begin{array}{ccccc|ccccc|c} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{6}$$

When we initialize into the $|0\rangle_L$ state, we would already have a logically encoded state (as all the Z-type stabilizers including Z_L should automatically be satisfied); however, these stabilizers are still measured anyway for a fault-tolerant initialization. In the case of transversal injection, Z_L is in some superposition of states, determined by the stabilizers that either commute or anticommute—our stabilizer trajectory.

The logical state of the block is $z_L \in \{0, 1\}$, where 0 indicates a $+1$ eigenstate of the logical Z operator ($|0\rangle_L$) and 1 as the -1 eigenstate ($|1\rangle_L$). The chain operator is also determined as

$$z_L = \left(\sum_{i \in \text{left/right}} z_i\right) \mod 2, \tag{7}$$

where left/right is the set of qubits of any chosen chain that runs from the left boundary of the lattice to the right (defining the logical Z operator). We perform a parity check over each qubit in the chain to give us z_L. If $z_L = 0$, we have initialized the logical qubit into the $|0\rangle_L$ state, and if $z_{\text{chain}} = 1$, the $|1\rangle_L$ state.

When all qubits are initialized into $|\chi\rangle^{\otimes N}$ before encoding, we are not simply in ± 1 eigenstates of Z_{p_i}. Instead, we are in a superposition of the 0 and 1 eigenstates.

For a more compact expression of the tableau in Eq. (5), we use the concatenated form H_x, H_z, $H_{Z_{\text{chain}}}$ (e.g., $H_{x_k} = x_{k,1} \ldots x_{k,N}$). Note that, in the surface code, the X-type vertex stabilizers and Z-type plaquette stabilizers are only populated with X and Z operators, respectively. Each possible stabilizer trajectory (indexed here by i) will determine a logical state $|\Lambda\rangle_i$ and will have the following form after encoding:

$$|\Lambda\rangle_i = \left[\begin{array}{cc} H_{x_1} & X_{v_1} \\ \vdots & \vdots \\ H_{x_{(N-1)/2}} & X_{v_{(N-1)/2}} \\ \hline H_{z_1} & Z_{p_1} \\ \vdots & \vdots \\ H_{z_{(N-1)/2}} & Z_{p_{(N-1)/2}} \\ H_{Z_{\text{chain}}} & Z_L \end{array}\right]_i = [\langle x_i, z_i, L_i \rangle]. \tag{8}$$

As the stabilizers of our code are constant, we can omit H_{x2}, $H_{x2} \ldots$ from our shorthand representation. What we are left with is x_i and z_i, the eigenvalue bit strings corresponding to the measurements of the stabilizers, and L_i, the eigenvalue bit string formed from the original $|\Lambda\rangle_i$ in each term, summed modulo 2.

Using the syndrome measurements, we know how the stabilizer sets transform from the original diagonal Z form to the stabilizers and logical operators for the encoded planar surface code. To calculate the resultant logical state, we now treat each term individually and follow them through the encoding procedure.

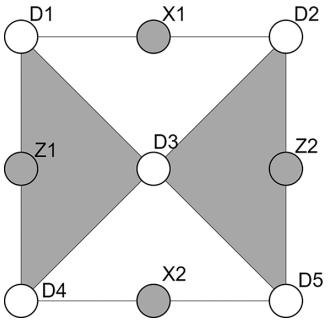

FIG. 1. Qubit layout of the unrotated surface code at distance 2.

C. Resultant state

As we are no longer in eigenstates of the logical Z operator of the surface code, we will be left with some nontrivial logically encoded state. The probabilistic nature of the X- and Z-type stabilizer measurements means that the resultant encoded state from transversal injection is probabilistic but heralded.

The fact that we follow only one of the $2^{(N-1)}$ possible stabilizer trajectories when we initialize the code means we select out only the terms that are consistent with the eigenvalues we measure. For example, in the $d = 2$ example in Fig. 1, if $Z_{p_1} = 0$, for stabilizer $Z_{p_1} = Z_{D1}Z_{D3}Z_{D4}$, then we can only keep the terms where $Z_{p_1} = (z_{D1} + z_{D3} + z_{D4}) \mod 2 = 0$. Any term where $Z_{p_1} = 1$ will be inconsistent with the measurement projection that actually occurred and will be simply projected out of the resultant state.

What we are left with is a superposition of only the eigenstates that are consistent with the X- and Z-stabilizer trajectories that are observed when initializing the state. By definition—as we have stabilized our data qubits with respect to all stabilizers of the planar surface code—we are going to be left with some superposition of the eigenstates of the logic operator Z_L, i.e., some new encoded state, $|\Lambda\rangle_i = \alpha_L|0\rangle_L + \beta_L|1\rangle_L$. Note that the resultant logical state does not, in general, match the transversal state $|\chi\rangle$ that was used to initialize each of the physical qubits in the encoded block.

To calculate the resultant logical state, we examine the last entry of the eigenvalue vector which is the logical observable $L_i = z_{\text{chain}}$ formed from the eigenvalues of the individual data qubits that form a connected left/right chain through the planar surface code. The amplitude of the resulting $|0\rangle_L$ state, α_L, will be the sum of all the terms where $L_i = 0$, while the amplitude of the $|1\rangle_L$, β_L, will be the sum of all the terms where $L_i = 1$. After renormalizing the wave function, we can have an analytical form of the resultant encoded state as a function of α, β, and N.

V. GENERAL ALGORITHM FOR CALCULATING THE FUNCTIONAL FORM OF INJECTED STATES

The above sections explain how analytical forms for the resultant logical state are calculated. We now want to present an explicit algorithmic construction that can be used to derive the logical state for an arbitrary distance d surface code.

If we are to only consider the trivial X-stabilizer trajectory ($|0\rangle$ syndromes in X), we find the four sets of analytical equations (up to renormalization) corresponding to the four sets of trajectories $\hat{t} = \{0000, 0001, 0010, 0011\}$. A worked example of $\hat{t} = \{0000\}$ can be found in the Appendix:

$$
\begin{aligned}
|\Lambda\rangle_{00} &= \begin{pmatrix}\alpha_L\\ \beta_L\end{pmatrix} = \begin{pmatrix}\alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4\\ 2\alpha^3\beta^2 + 2\alpha^2\beta^3\end{pmatrix},\\
|\Lambda\rangle_{11} &= \begin{pmatrix}\alpha_L\\ \beta_L\end{pmatrix} = \begin{pmatrix}\beta^5 + 2\beta^2\alpha^3 + \beta\alpha^4\\ 2\beta^3\alpha^2 + 2\beta^2\alpha^3\end{pmatrix},\\
|\Lambda\rangle_{01} = |\Lambda\rangle_{10} &= \begin{pmatrix}\alpha_L\\ \beta_L\end{pmatrix} = \begin{pmatrix}\alpha^4\beta + \alpha^3\beta^2 + \alpha^2\beta^3 + \alpha\beta^4\\ \alpha^4\beta + \alpha^3\beta^2 + \alpha^2\beta^3 + \alpha\beta^4\end{pmatrix}.
\end{aligned}
\tag{9}
$$

For two (out of four) measured trajectories, we get a nontrivial set of logical states ($\hat{t} = \{0000, 0011\}$). For the other two stabilizer trajectories ($\hat{t} = \{0001, 0010\}$), we simply initialize into the logical $|+\rangle_L = (|0\rangle_L + |1\rangle_L)/\sqrt{2}$ state.

It should be noted that any nontrivial syndrome will introduce bit and/or phase flips to the logical code words. As is typical of a stabilizer code, corrections need to be applied to move back into an even superposition of $|0\rangle_L$ and $|1\rangle_L$.

This generalized algorithm will allow for the calculation of any transversely realizable encoded state for an arbitrarily large distance code d: it intuitively follows the logic of the stabilizer execution.

Line 5 requires an explicit calculation of all vectors generated from the X-stabilizer projections and will scale exponentially as a function of code distance d. This proves to be a difficult task for classical computers, as there are an exponential number of trajectories to track during the execution of the algorithm. Additionally, the solution space is large, and storing the final results would be infeasible for large distances.

There is an optimized version of this algorithm, which we introduce in the Appendix. The optimized algorithm is based on a different approach and instead tracks only the terms relevant to a single target trajectory. We can now compute the logical state of this trajectory in a more efficient manner. Algorithm 2 scales exponentially with $(N-1)/2$, the number of Z stabilizers, rather than with the number of physical qubits. Hence, calculating a target trajectory is $O\{\exp[(N-1)/2]\}$, which is an improvement from $O(e^N)$ required to derive all possible trajectories. This improvement is most noticeable in the memory requirements which blow out in the first algorithm but can be dealt with comfortably in Algorithm 2. Using GPU acceleration, we have demonstrated this algorithm on trajectories with trivial X measurements on systems with over 100 data qubits. When nontrivial X measurements are considered, classical computation of distance 5 is achievable using Algorithm 2. This opens up the opportunity for a just-in-time strategy where output states from transversal injection can be calculated classically and then used as part of a broader compilation strategy. To realize this strategy, a more efficient algorithm will be needed for larger distances required for large-scale, error-corrected computation.

In Fig. 2, we have plotted the distribution of resultant states across the Bloch sphere for a range of initial parameters. There are clear structures that form depending on the parameters of the initial rotation including great circles, arcs with different radii, clusters around poles, and dense packing of the entire sphere. Certain distributions may be advantageous for certain compilation strategies or for physical systems that have error biases in certain directions. Alternatively, having a dense covering of the Bloch sphere might be optimal in a compilation strategy that benefits from a large set of distinct states.

VI. REALIZING NON-CLIFFORD LOGIC OPERATIONS THROUGH TRANSVERSAL INJECTION

The primary benefit of this technique is producing logically encoded non-Pauli states with a higher fidelity directly onto the surface code. This aims to reduce the qubit and time resources required to distill magic states and, in turn, the quantity of costly non-Clifford logic gates.

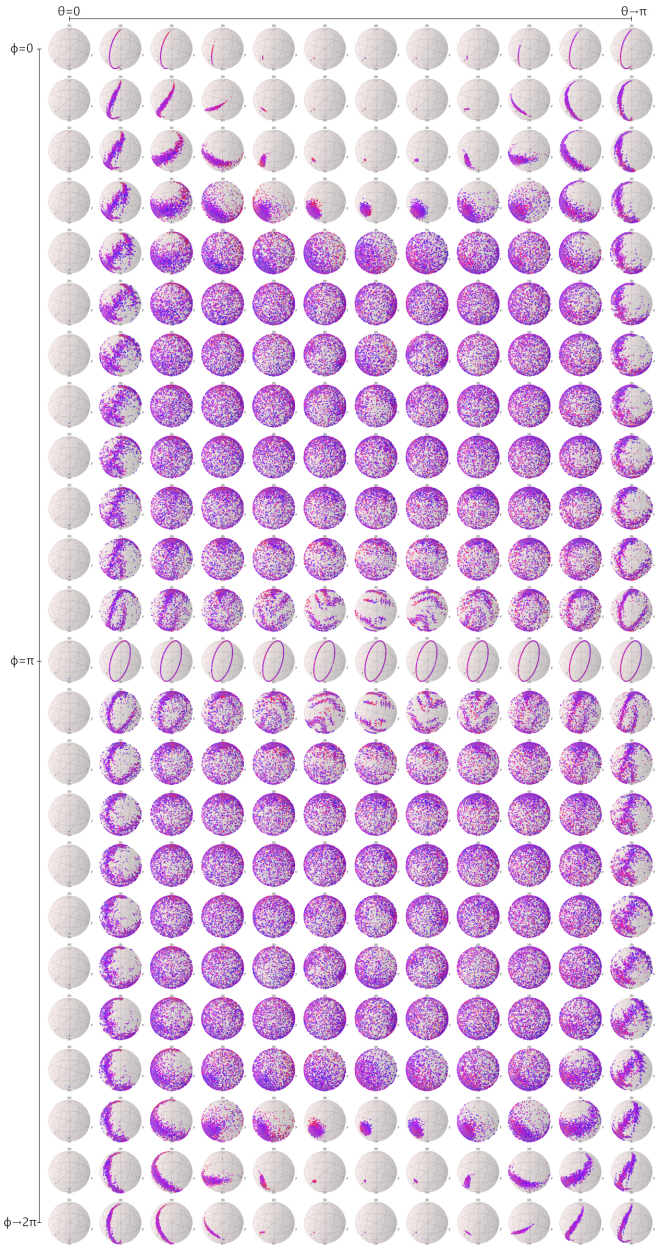

FIG. 2. The possible logical states that result from transversal injection on a distance $d = 4$ code for a range of initial θ and ϕ values. Each sphere is a plot for a specific initial rotation. Each dot on the sphere corresponds to a possible stabilizer trajectory and its respective output state. Generated using QuTiP [32].

Consider the circuits shown in Fig. 3. In each of these circuits, we have our data qubit $|\psi\rangle$ and an ancilla qubit that is prepared in the states $(|0\rangle + e^{i\theta}|1\rangle)/\sqrt{2}$ or $\cos(\theta)|0\rangle + i\sin(\theta)|1\rangle$. The rest of the circuit consists of the Clifford gates, CNOT, and measurements in either the X or Z basis. It can be easily verified that, after the ancilla is measured, the resultant state is the rotated gate $R_z(\pm\theta)$ or $R_x(\pm\theta)$ applied to the data qubit $|\psi\rangle$. The angle θ is determined by the form of the ancillary state, and the $\pm$ is determined by the outcome of the measurement result on the ancilla. Consequently, the ability to perform arbitrary single-qubit rotations becomes a problem of preparing appropriate ancillary states.

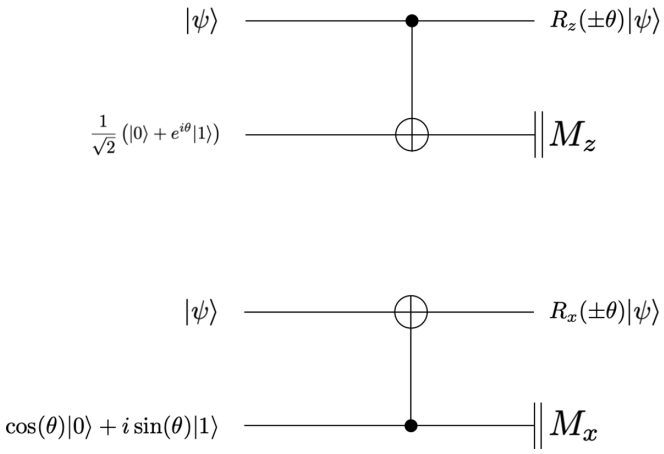

FIG. 3. Quantum circuits to use non-Clifford states to enact non-Clifford gates.

To match up the functional forms in Eq. (9) to the functional forms necessary for the circuits in gate teleportation, in the context of our $d = 2$ example, we need to solve the following:

$$\begin{pmatrix} 1 \\ e^{i\theta} \end{pmatrix} = \begin{pmatrix} \alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4 \\ 2\alpha^3\beta^2 + 2\alpha^2\beta^3 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} \beta^5 + 2\beta^2\alpha^3 + \beta\alpha^4 \\ 2\beta^3\alpha^2 + 2\beta^2\alpha^3 \end{pmatrix}$$

and

$$\begin{bmatrix} \cos(\theta) \\ i\sin(\theta) \end{bmatrix} = \begin{pmatrix} \alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4 \\ 2\alpha^3\beta^2 + 2\alpha^2\beta^3 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} \beta^5 + 2\beta^2\alpha^3 + \beta\alpha^4 \\ 2\beta^3\alpha^2 + 2\beta^2\alpha^3 \end{pmatrix}, \tag{10}$$

for α, β, and θ (up to renormalization), in the case of the $d = 2$ solution. Once we find specific forms of α and β that give rise to these functional forms, we know what transversal injected states are needed on the physical qubits to generate the correct form of the encoded states to be used in the circuits in Fig. 3 to realize $R_z(\theta)$ and $R_x(\theta)$ rotational gates.

A. Eastin-Knill theorem

Transversal injection circumvents the Eastin-Knill no-go theorem, as it is a method for preparing non-Clifford resources, such as a T-gate, to achieve universal quantum computation on the planar surface code. This technique does not violate the Eastin-Knill theorem and is in fact still limited by the implications of the theorem. Non-Clifford gates must be realized in the fashion described in the preceding section; a T-gate still cannot be implemented transversally on this code. Additionally, the encoding step in this technique has a diminished error-detecting ability compared with standard surface code operation. Some errors in the first round of stabilizer measurements can no longer be detected. Such errors will herald an incorrect stabilizer trajectory which implies that the system is in a different logical state than it actually is. Postselection has the potential to mitigate this. However, we are still ultimately bound by this theorem.

VII. NUMERICAL SIMULATIONS OF ERROR SCALING

Typically, an encoded state produced using the planar surface code will exhibit asymptotic fault tolerance. Transversal injection does not introduce any new multiqubit gate operations beyond what is required for standard $|0\rangle_L$ and $|+\rangle_L$ state initialization, so one may expect the same scaling out of this protocol. We confirm this assumption with numerical simulation where errors occur only after encoding; however, this scenario is unrealistic, and the protocol as a whole must be considered.

The circuit is simulated by applying a depolarizing channel to the circuit. At a physical rate p, a Pauli X, Z, or Y operator is applied with equal probability for a single-qubit gate. For the two-qubit CNOT gate for syndrome extraction and measurement, a random combination of I, X, Z, and Y operators (excluding II) is applied. The simulations shown in this paper are performed with physical errors during all rounds of stabilizers, including the first stabilizer measurements. This generally results in an encoded error rate greater than the physical error rate p. However, experiments with postselection strategies indicate that an error rate not bound by p is in fact achievable, warranting further research into the behavior of transversal injection at distance 5 and higher.

To test transversal injection, surface code circuit simulations were performed using a balanced Pauli noise model. As we need to verify a nontrivial final state against the output from the derived logical state above, we used a full state simulator rather than stabilizer simulators such as Stim [33]. The circuit simulations were performed with d rounds of stabilizer measurements followed by a final perfect round of measurements. Where the syndromes have changed, the series of measurements observed can be mapped to a correction operator and applied to correct back into the expected resultant state. Discarding states that have detectable errors will reduce the success probability of the protocol compared with a strategy that instead corrects errors that have been detected. For large code distances or particularly high physical error rates, this may not be a feasible strategy, although for the error rates and distances investigated in this paper, this is not the case. Additionally, applying the error-correction operator can introduce further errors, so qualitatively, the fidelity can only decrease during this process. Hence, for all numerical data in this paper, any simulations where the syndromes change over multiple sweeps are discarded.

The unrotated surface code was used for the worked examples and for initial testing. However, the rotated surface code was subsequently chosen for numerical simulations, as the lower number of qubits allowed a statistically significant amount of data to be acquired (Fig. 4). Additionally, the numerical data presented in this paper were collected by reusing ancilla qubits for syndrome extraction. Since we are not constricted to nearest-neighbor architecture in classical simulations, this allows us to profile transversal injection up to distance 4 across different uniform physical error levels and differing single-/two-qubit error rates.

First, a transversal rotation of all data qubits is performed. Ancillary qubits perform CNOT operations on their nearest neighbors in the X and Z bases for the vertex and plaquette stabilizers, respectively. The ancillary qubits are measured, extracting the syndrome as is standard for the surface code (Fig. 5). The syndrome measurements and parameters of the transversal rotation are passed into Algorithm 2, where the state of our system is returned in an analytical form. The full state of our simulated system is yielded and compared with the result from Algorithm 2 to determine if a logical error has occurred.

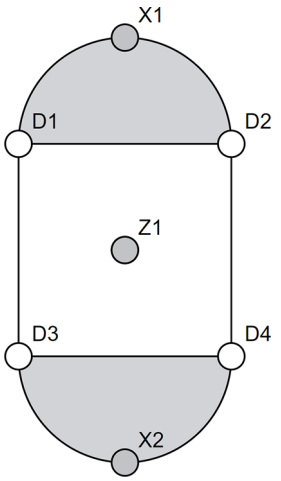

FIG. 4. Qubit layout of the rotated surface code at distance 2.

Fidelity appeared to vary for different initial states, and a chosen state too close to a pole on the Bloch sphere would snap to that pole for a large portion of the trajectories. Hence, the number of unique non-Clifford logical states seems to increase proportionately to how far away an initial rotation is from the poles.

The relationship between physical error rate and logical rate are linearly proportional, as we approach physical error rates below $p = 0.01$ (Fig. 6). It appears that higher distance codes perform worse than lower distance codes, and the logical error rate always exceeds the physical error rate. Without a postselection strategy, numerical simulations can be done more efficiently at lower physical error rates.

For each distance and physical error rate, a second experiment was run where a postselection strategy was employed in an attempt to improve fidelity (Fig. 7). Postselection appears

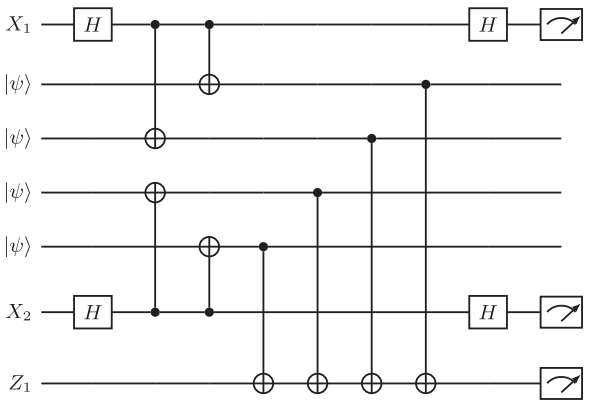

FIG. 5. Circuit diagram of the rotated surface code at distance 2.

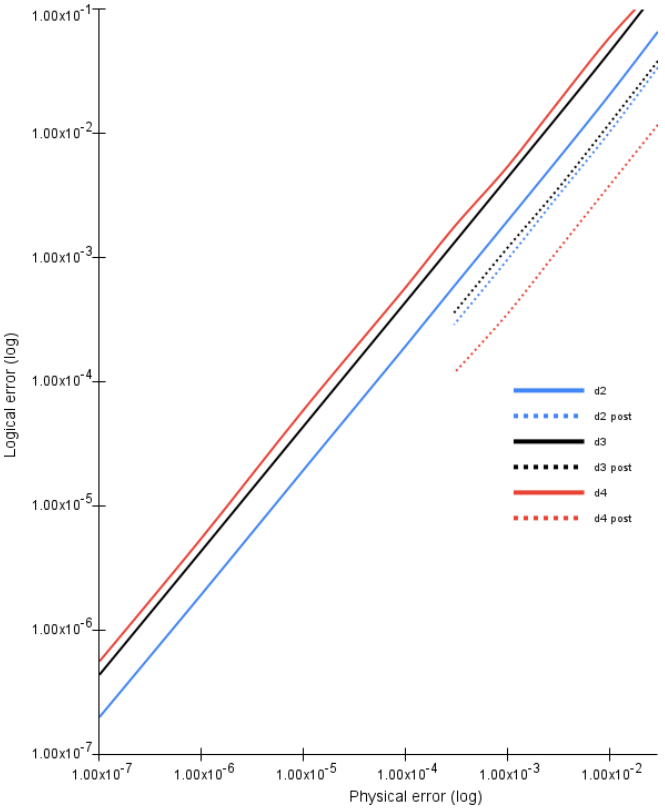

FIG. 6. Scaling of logical fidelity vs physical error rate. Fidelity is measured as the overlap between the simulated encoded state and the expected encoded state heralded by the stabilizer measurements of each run. $\theta_{\text{initial}} \approx 1.7728$, $\phi_{\text{initial}} \approx 3.3237$.

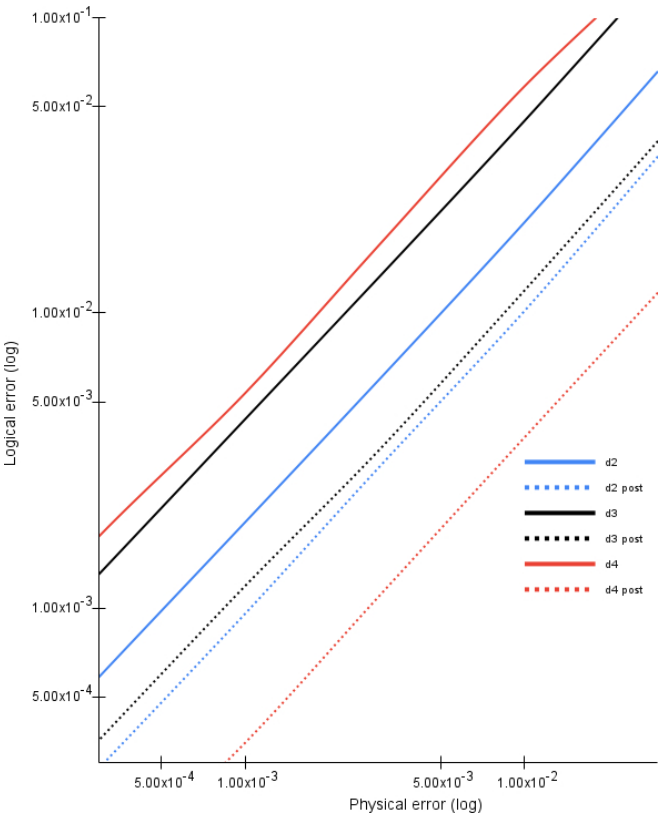

FIG. 7. This figure is the same set of results as Fig. 6, inspecting a smaller range of physical error rates to focus on the postselection results. $\theta_i \approx 1.7728$, $\phi_i \approx 3.3237$.

to yield improvements in fidelity by filtering out trajectories of simulated runs with a naïve, precomputed lookup table. For distance 2, this lookup table was only populated with the trivial syndrome, as it is the only one that results in non-Clifford states. Distance 4 with postselection appears to yield a significant improvement in fidelity, dropping below the physical error rate to roughly $0.39p$. All of the above experiments were benchmarked with nonuniform single-qubit error rates of p, $0.1p$, and $0p$ with no measurable difference in fidelity.

To construct the lookup table used for an experiment, statistical analysis was done over a large number of simulations, and the trajectories were ranked according to their average fidelity. A budget of ≈20% was allocated, and the top trajectories were selected until this budget was satisfied. An example for demonstration (not experimental data) is in Table I. Given these data, we would whitelist the top performing trajectories (000, 011, and 101) until our ≈20% quota was satisfied.

VIII. CONSEQUENCES FOR ERROR-CORRECTED CIRCUIT COMPILATION

Transversal injection has the potential to reduce ancilla requirements and the amount of state distillation; however, it is not without its own drawbacks. Transversal injection provides us with non-Clifford, logically encoded states after the first round of stabilizers are applied. The first round of stabilizers themselves are not fault tolerant, and this round is susceptible to two-qubit correlated errors without a detection event. If these errors occur before the first stabilizer measurements are extracted, the initial state will be altered without detection events. Single-qubit errors on ancillary qubits are either detected in subsequent syndrome extractions or change the logical state in a way that is heralded by its measurement (i.e., once the initial Pauli frame of an encoded state is known, the encoded state is defined).

Current methods of magic state encoding similarly exhibit a linear scaling between the fidelity of the encoded state and physical error rate [30]. Generally, the fidelity of output states from transversal injection is worse until a postselection strategy is applied. Once postselection is applied on our distance

TABLE I. Example for demonstrating a postselection strategy.

Trajectory	Fidelity	Frequency
011	99.99	0.01
000	99.89	0.10
101	98.58	0.09
001	98.01	0.20
100	97.84	0.31
010	95.43	0.20
110	95.21	0.05
111	94.99	0.04

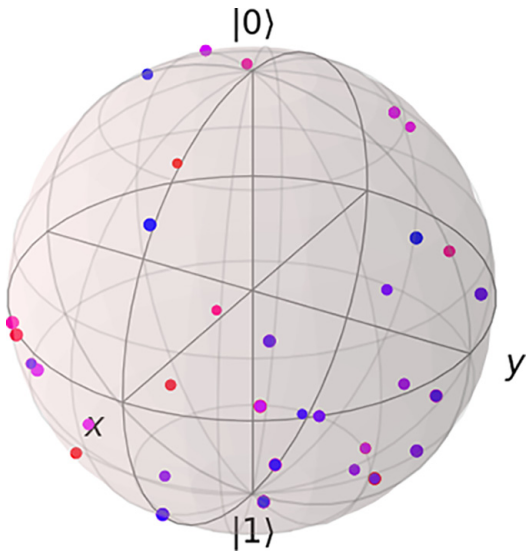

FIG. 8. The 64 logical, single-qubit states (only trivial X-stabilizer measurements), resultant from transversal injection on a distance $d = 3$ unrotated code from the table above. Generated using QuTiP [32].

4 simulations, the logical error rate is comparable with the lower bound of other results, even at much larger distances. To achieve the same fidelity, transversal injection has a much smaller qubit footprint and only requires a moderate amount of postselection. Further analysis is needed to evaluate this protocol at distance 5 and higher to determine how fidelity scales beyond the results in this paper.

Transversal injection is intrinsically a probabilistic method for encoded state preparation, and while it is heralded, it is not something that is controllable. Consequently, when utilizing transversal injection as a method for realizing universality, special care must be taken when examining how to effectively compile error-corrected circuits.

While many states on the logical Bloch sphere are available using this technique, the number of possible stabilizer trajectories scales exponentially as a function of the number of qubits and hence code distance. In our preliminary analysis, the probability of a particular state being prepared becomes exponentially unlikely as the code distance is scaled. There are potential redundancies at higher distances, but so far, we have not identified any exploitable patterns.

This implies that, when a specific ancillary state is required for a teleported gate, it will need to be constructed through an effective random walk over the Bloch sphere. The exponential number of states on the Bloch sphere that is available implies that, rather than compiling to simply the non-Clifford T-gate in an error-corrected system, we should be able to compile to any Z- or X-axis rotation we want by producing random logical states and approximating the required ancillary state needed for a given $R_z(\theta)$ or $R_x(\theta)$. Transversal injection can be used to produce magic states which we can then distill to an arbitrary accuracy, although this is only a subset of the possible output states. Hence, there is motivation to research distillation methods that are effective on other non-Pauli states. This has clear implications for circuit-level compilation, as the Clifford + T alphabet for a fault-tolerant compatible circuit will no longer be a constraint.

There are various techniques developed in the literature in approximating single-qubit gates via a random walk around SU(2) that can be exploited to find a systematic solution to the gate compilation issue [25], but this is relegated to further work.

While a direct resource comparison with a compiled algorithm using magic state distillation, such as Shor-2048 [15], will require a systematic solution to compiling arbitrary single-qubit logical rotations, there is a potential for reduced qubit requirements for any large-scale algorithm by utilizing this technique.

IX. CONCLUSIONS

We have presented a technique for achieving fault-tolerant universal quantum computation in a stabilizer code environment without changing any other operating assumption of the underlying code.

While we have presented this scheme in the context of the surface code, transversal injection will be possible using any stabilizer-based QEC code, and the algorithm detailed in this paper can be used to precompute resultant logical states, depending on the code structure and size.

Interesting further work includes understanding if this technique can be used for codes beyond stabilizer codes and how transversal injection can be used in fully compiled error-corrected algorithms. Incorporating transversal injection into fully compiled, large-scale circuits will allow us to rebenchmark algorithms such as Shor's algorithm or error-corrected chemistry simulations to determine the exact resource savings over state-of-the-art compiled results.

It should be noted that the algorithms presented in this paper scale exponentially with the number of Z stabilizers. Currently, we can compute explicit analytical forms for any stabilizer trajectory up to a $d = 8$ nonrotated surface code (consisting of $N = 113$ data qubits). This will be more than sufficient for any experimental demonstration in the near term. With further development and specialized hardware, higher distances will likely be possible to calculate.

However, we anticipate that a more efficient technique can be developed that allows for analytical forms to be calculated fast. This may not allow for all of the exponential number of trajectories to be precalculated, but if a single trajectory for an arbitrary distance can be calculated fast, then this will be sufficient for a just-in-time approach to be taken, where analytical forms for logical states are calculated at the time they are physically created in an error-corrected machine.

In this paper, we offer a potential solution to some of the overheads blocking the path to large-scale, error-corrected computation and provide another approach to circumvent the Eastin-Knill theorem, allowing for universal, error-corrected computation in an increasingly resource-efficient manner.

ACKNOWLEDGMENTS

Thank you to Austin Fowler for early feedback on the findings of this paper and Sam Elman for help with editing the manuscript. The views, opinions, and/or findings expressed

are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. This research was developed in part with funding from the Defense Advanced Research Projects Agency [under the Quantum Benchmarking (QB) program under Award No. HR00112230007 and HR001121S0026 contracts]. M.J.B. acknowledges the support of Google. M.J.B., J.G., and A.S. were supported by the ARC Centre of Excellence for Quantum Computation and Communication Technology (CQC2T), Project No. CE170100012. A.S. was also supported by the Sydney Quantum Academy.

APPENDIX A: WORKED EXAMPLE OF ALGORITHM 1

For example, in a distance $d = 2$ unrotated code (Fig. 1), we have a total of $N = 5$ physical qubits. The parity check matrix M is a 5×5 matrix with rows representing the two X stabilizers, two Z stabilizers, and one logical operator of the distance two surface code:

$$M = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}. \tag{A1}$$

The initialization step will take us to the following state:

$$|\chi\rangle^{\otimes N} = \sum_{n=0}^{2^N-1} \alpha^{N-\hat{H}(n)} \beta^{\hat{H}(n)} |n\rangle . \tag{A2}$$

Let us work out the analytical equations for the trivial syndrome. Below is a partial state table of our initial state. Now we project onto X-type stabilizers $XXXII$ and then $IIXXX$ in the middle and right columns, respectively:

$$\left(\begin{array}{c} \\ |00000\rangle \\ |00001\rangle \\ |00010\rangle \\ |00011\rangle \\ |00100\rangle \\ |00101\rangle \\ \cdots \\ |11010\rangle \\ |11011\rangle \\ |11100\rangle \\ |11101\rangle \\ |11110\rangle \\ |11111\rangle \end{array}\right) = \left(\begin{array}{c|c|c} \text{init} & XXXII & IIXXX \\ \alpha^5 & \alpha^5 + \alpha^2\beta^3 & \alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4 \\ \alpha^4\beta & \alpha^4\beta + \alpha\beta^4 & \alpha^4\beta + \alpha^3\beta^2 + \alpha^2\beta^3 + \alpha\beta^4 \\ \alpha^4\beta & \alpha^4\beta + \alpha\beta^4 & \alpha^4\beta + \alpha^3\beta^2 + \alpha^2\beta^3 + \alpha\beta^4 \\ \alpha^3\beta^2 & \alpha^3\beta^2 + \beta^5 & \alpha^4\beta + 2\alpha^3\beta^2 + \beta^5 \\ \alpha^4\beta & \alpha^4\beta + \alpha^3\beta^2 & \alpha^4\beta + 2\alpha^3\beta^2 + \beta^5 \\ \alpha^3\beta^2 & \alpha^3\beta^2 + \alpha^2\beta^3 & \alpha^4\beta + \alpha^3\beta^2 + \alpha^2\beta^3 + \alpha\beta^4 \\ \cdots & \cdots & \cdots \\ \alpha^2\beta^3 & \alpha^3\beta^2 + \alpha^2\beta^3 & \alpha^4\beta + \alpha^3\beta^2 + \alpha^2\beta^3 + \alpha\beta^4 \\ \alpha\beta^4 & \alpha^2\beta^3 + \alpha\beta^4 & \alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4 \\ \alpha^2\beta^3 & \alpha^5 + \alpha^2\beta^3 & \alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4 \\ \alpha\beta^4 & \alpha^4\beta + \alpha\beta^4 & \alpha^4\beta + \alpha^3\beta^2 + \alpha^2\beta^3 + \alpha\beta^4 \\ \alpha\beta^4 & \alpha^4\beta + \alpha\beta^4 & \alpha^4\beta + \alpha^3\beta^2 + \alpha^2\beta^3 + \alpha\beta^4 \\ \beta^5 & \alpha^3\beta^2 + \beta^5 & \alpha^4\beta + 2\alpha^3\beta^2 + \beta^5 \end{array}\right). \tag{A3}$$

When the Z-type stabilizers are measured, the terms that do not satisfy the parity check are now excluded from the state table. This gives us our resultant state:

$$\begin{pmatrix} |00000\rangle \\ |00111\rangle \\ |01001\rangle \\ |01110\rangle \\ |10010\rangle \\ |10101\rangle \\ |11011\rangle \\ |11100\rangle \end{pmatrix} = \begin{pmatrix} \alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4 \\ \alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4 \\ 2\alpha^3\beta^2 + 2\alpha^2\beta^3 \\ 2\alpha^3\beta^2 + 2\alpha^2\beta^3 \\ 2\alpha^3\beta^2 + 2\alpha^2\beta^3 \\ 2\alpha^3\beta^2 + 2\alpha^2\beta^3 \\ \alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4 \\ \alpha^5 + 2\alpha^2\beta^3 + \alpha\beta^4 \end{pmatrix}. \tag{A4}$$

APPENDIX B: OPTIMIZATION OF ALGORITHM 1

We can provide this refinement to Algorithm 1 as detailed below.

In this more performant algorithm, the goal is to determine which eigenvalues commute with the Z sector of our trajectory and project these onto the target X sector. The original algorithm projects our initial state through the X stabilizers, which has an exponential number of terms in d, and only a fraction contribute to our encoded logical state for a specific Z trajectory. By working backwards, we only need to reverse engineer the terms that will commute with our Z stabilizer measurements. The Z-trajectory search is done in the loop from lines 3 to 19. For each Z stabilizer, we can search for the combinations of the bits $= Z$ that XOR to give the correct value for our target.

For example, consider a distance 2 unrotated surface code (Fig. 1) with 5 data qubits and a target $00\hat{0}1$. The two Z stabilizers of the distance two surface code form the matrix below:

$$M = \begin{pmatrix} D_1 & D_2 & D_3 & D_4 & D_5 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}. \tag{B1}$$

If we create a representation of the Z stabilizers where we store the index of elements equal to one, we have a directory of which data qubits each Z stabilizer impacts. In this example,

Algorithm 1 Calculating a resultant logical state.

Require: $X_1 \ldots X_{(N-1)/2}$, X stabilizers.
Require: $Z_1 \ldots Z_{(N-1)/2}$, Z stabilizers.
Require: L, a chain operator for the logical Z state.
Require: $\{\alpha,\beta\}$, coefficients of transversal physical states.
Require: Number of data qubits of a distance d planar surface code, unrotated code, $N = d^2 + (d-1)^2$.

function Loop($\langle x_i\rangle \leftarrow \hat{x}$) # For each X stabilizer trajectory
  function Loop($n \leftarrow 1$ to $2^N - 1$) # For each eigenstate n
    j = Hamming(n)
    $\hat{k} = |n\rangle \times \langle x_i|$ # Project into eigenstates of X stabilizers
    $\lambda(\hat{k}) = j$ # Dictionary of original Hamming weight
  end function
  function Loop($\langle z_i\rangle \leftarrow \hat{z}$) # For each Z stabilizer trajectory
    $\alpha_L(x_i, z_i) = \beta_L(x_i, z_i) = 0$
    function Loop($k_i \leftarrow \hat{k}$) # For each eigenstate n
      $m = \hat{k} \times \hat{z}$ # Parity check with Z stabilizers
      if $\boldsymbol{m \neq 0}$
      $j = \lambda(\hat{k})$
      $l = \hat{k} \times B$ # Parity check with logical operator
      if $l = 0$, $\alpha_L(x_i, z_i) = \alpha_L(x_i, z_i) + \alpha^j \beta^{N-j}$.
      if $l = 1$, $\beta_L(x_i, z_i) = \beta_L(x_i, z_i) + \alpha^j \beta^{N-j}$.
    end function
  end function
  return $\{\alpha_L(\hat{z}), \beta_L(\hat{z})\}/\sqrt{|\alpha_L(\hat{z})|^2 + |\beta_L(\hat{z})|^2}$
end function

we use 0 as the first index:

$$M_{\text{aux}} = \begin{pmatrix} 0 & 2 & 3 \\ 1 & 2 & 4 \end{pmatrix}. \tag{B2}$$

For the first Z stabilizer, we look up the first bit of $\hat{01}$, which is 0. Consider all combinations of $\{0, 2, 3\}$ that are of length $\{0, 2\}$ (or $\{1, 3\}$ if it were equal to 1). We now iterate through each of $\{(\_), (0, 2), (0, 3), (2, 3)\}$ and set the bits of our initial trajectory $\{0\}_N$:

$$\hat{v_t} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}. \tag{B3}$$

For each subsequent Z stabilizer, we set t equal to the ith bit of $\hat{01}$. Depending which values of $\{1, 2, 4\}$ have been set already, we create two subsets for the seen and unseen indices (i.e., $\{2\}$ and $\{1, 4\}$). For each result from the previous loop, we **xor** t with the value of each bit at the seen indices. As in the first case, we iterate through all combinations of unseen bits $\{1, 4\}$ that are odd or even lengths depending on t. Again, we set the bits of the result at the indices—for each new combination, for each result from the previous loop. In our example below, t will alternate between 0 and 1, as z_2 is our unseen, and as t is initialized as 1 for the second bit, results will be flipped:

$$\left[\begin{array}{c|ccccc|c} t & z_0 & z_1 & z_2 & z_3 & z_4 & \text{combinations} \\ 1 & 0 & 0 & 0 & 0 & 0 & (Z_{p_1}), (Z_{p_4}) \\ 0 & 1 & 0 & 1 & 0 & 0 & (\_), (Z_{p_1}, Z_{p_4}) \\ 1 & 1 & 0 & 0 & 1 & 0 & (Z_{p_1}), (Z_{p_4}) \\ 0 & 0 & 0 & 1 & 1 & 0 & (\_), (Z_{p_1}, Z_{p_4}) \end{array}\right], \tag{B4}$$

$$\text{Results} = \left[\begin{array}{ccccc|c} z_0 & z_1 & z_2 & z_3 & z_4 & j \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 2 \\ 1 & 1 & 1 & 0 & 1 & 4 \\ 1 & 0 & 0 & 1 & 1 & 3 \\ 1 & 1 & 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 & 1 & 4 \end{array}\right]. \tag{B5}$$

Algorithm 2 Calculating a specific logical state.

Require: $X_1 \ldots X_{(N-1)/2}$, X stabilizers.
Require: $Z_1 \ldots Z_{(N-1)/2}$, Z stabilizers.
Require: L, a chain operator for the logical Z state.
Require: $\{\alpha,\beta\}$, coefficients of transversal physical states.
Require: Number of data qubits of a distance d planar surface, unrotated code, $N = d^2 + (d-1)^2$.
Require: $\langle x_i\rangle$, $\langle z_i\rangle$ eigenvalue bit strings corresponding to stabilizer measurements.

$\hat{m} = \{\}$ # List of set stabilizer measurements $(N-1)/2$
$\hat{v} = [\{0\}_N]$ # Eigenvalue memory, initially just $\{0\}_N$
function Loop($i \leftarrow 1$ to $(N-1)/2$) # For each Z stabilizer
  $\hat{v_t} = [\,]$ # Loop eigenvalue storage
  function Loop($v \leftarrow \hat{v}$) # For each previous result
    $t = z_i$
    function Loop($m \leftarrow \hat{m}$)
      $t \oplus (m_i)$
    end function
    $\hat{u} = \neg\hat{m}$ # can ignore Z_i not adjacent to ith qubit
    $\hat{c} = \sum_{n=0} \binom{\hat{u}}{2n+t}$ # combinations that give parity t
    $\hat{m} = \hat{m} + Z_i$
    function Loop($c \leftarrow \hat{c}$)
      $\hat{e} = \hat{z} \vee c$
      $\hat{v_t} = \hat{v_t} + e$
    end function
    $\hat{v} = \hat{v_t}$
  end function
end function
$\alpha_L = \beta_L = 0$
function Loop($v \leftarrow \hat{v}$)
  $j = bitwise\_sum(\hat{z})$
  $\hat{k} = \hat{v} \times \hat{x}$
  function Loop($k \leftarrow \hat{k}$)
    $l = k \times B$ # Parity check with logical operator
    if $l = 0$, $\alpha_L = \alpha_L + \alpha^j \beta^{N-j}$.
    if $l = 1$, $\beta_L = \beta_L + \alpha^j \beta^{N-j}$.
  end function
end function
return $\{\alpha_L, \beta_L\}/\sqrt{|\alpha_L|^2 + |\beta_L|^2}$

TABLE II. Stabilizer trajectories and the corresponding resultant state in the distance three surface code for a specific initial state ($\theta = 2.445\,805\,631\,497\,81$, $\phi = 1.361\,697\,088\,568\,559\,5$). The rows highlighted in bold are resource states for implementing a T-gate.

Z-traj	θ_L	ϕ_L
000$\hat{0}$000	1.477251531844677	−2.286354913393752
000$\hat{0}$001	2.4598774702571236	−2.020056925558729
000$\hat{0}$010	2.014303116800432	−0.37430535389058534
000$\hat{0}$011	2.0000126349446785	1.0875192568311256
000$\hat{1}$100	2.4598774702571236	−2.020056925558729
000$\hat{1}$101	2.2207374531113	2.2914520972008927
000$\hat{1}$110	2.0000126349446785	1.0875192568311256
000$\hat{1}$111	1.5899082238746929	−0.7176718165630172
001$\hat{0}$000	0.6817151833326703	2.020056925558729
001$\hat{0}$001	2.6105434793012248	2.3851586668749465
001$\hat{0}$010	0.6010286049447089	2.066850378564112
001$\hat{0}$011	1.505967601101266	0.5553720557783275
001$\hat{1}$100	2.3652945274675323	−1.1917847714989986
001$\hat{1}$101	0.4526355842480339	−2.1740432175058464
001$\hat{1}$110	2.0155964891204774	−1.1347363564914656
001$\hat{1}$111	2.0275033715124824	0.5744495240028695
010$\hat{0}$000	1.127289536789362	0.37430535389058534
010$\hat{0}$001	0.6010286049447089	2.066850378564112
010$\hat{0}$010	1.113903157150333	1.5597871287260436
010$\hat{0}$011	**1.5707963267679674**	**−0.7853981633911511**
010$\hat{1}$100	0.6010286049447089	2.066850378564112
010$\hat{1}$101	0.8387195533772056	−2.58989408067244
010$\hat{1}$110	**1.5707963267679674**	**−0.7853981633911511**
010$\hat{1}$111	1.791506609499127	0.6159885397597338
011$\hat{0}$000	2.0000126349446785	1.0875192568311256
011$\hat{0}$001	1.6356250524885274	−0.5553720557783275
011$\hat{0}$010	**1.5707963268218257**	**0.7853981633911511**
011$\hat{0}$011	2.2207374531113	2.2914520972008927
011$\hat{1}$100	1.125996164469316	1.1347363564914656
011$\hat{1}$101	2.316857739052819	−2.723267254060598
011$\hat{1}$110	1.375057538833144	1.7128079495289403
011$\hat{1}$111	2.2924864906921796	0.023764711852680698
100$\hat{0}$000	0.6817151833326703	2.020056925558729
100$\hat{0}$001	2.3652945274675323	−1.1917847714989986
100$\hat{0}$010	0.6010286049447089	2.066850378564112
100$\hat{0}$011	2.0155964891204774	−1.1347363564914656
100$\hat{1}$100	2.6105434793012248	2.3851586668749465
100$\hat{1}$101	0.4526355842480339	−2.1740432175058464
100$\hat{1}$110	1.505967601101266	0.5553720557783275
100$\hat{1}$111	2.0275033715124824	0.5744495240028695
101$\hat{0}$000	2.2207374531113	2.2914520972008927
101$\hat{0}$001	2.68895706934176	2.174043217505847
101$\hat{0}$010	2.3028731002125875	2.58989408067244
101$\hat{0}$011	2.316857739052819	−2.723267254060598
101$\hat{1}$100	2.68895706934176	2.174043217505847
101$\hat{1}$101	2.760788786089281	−1.1274336283037671
101$\hat{1}$110	2.316857739052819	−2.723267254060598
101$\hat{1}$111	2.370102532312507	1.9493577457230364
110$\hat{0}$000	2.0000126349446785	1.0875192568311256
110$\hat{0}$001	1.125996164469316	1.1347363564914656
110$\hat{0}$010	**1.5707963268218257**	**0.7853981633911511**
110$\hat{0}$011	1.375057538833144	1.7128079495289403
110$\hat{1}$100	1.6356250524885274	−0.5553720557783275
110$\hat{1}$101	2.316857739052819	−2.723267254060598
110$\hat{1}$110	2.2207374531113	2.2914520972008927
110$\hat{1}$111	2.2924864906921796	0.023764711852680698
111$\hat{0}$000	1.5516844297151002	0.717671816563017
111$\hat{0}$001	2.0275033715124824	0.5744495240028695
111$\hat{0}$010	1.791506609499127	0.6159885397597338
111$\hat{0}$011	0.8491061628976135	−0.023764711852680698
111$\hat{1}$100	2.0275033715124824	0.5744495240028695
111$\hat{1}$101	0.7714901212772858	−1.9493577457230364
111$\hat{1}$110	0.8491061628976135	−0.023764711852680698
111$\hat{1}$111	1.7094217253738777	2.868327212586131

For each row, we determine Hamming weight j from the count of ones in $\hat{z}$. Note j will remain the same during the next step even though the Hamming weight will change.

It should be noted that, after any loop in the algorithm, the intermediate results can be processed in parallel with no penalty in complexity aside from the negligible overhead of distributing the work. Using GPU acceleration, we have demonstrated sampling from the set of all trajectories with trivial X syndromes for stabilizer codes with 113 data qubits ($d = 8$, unrotated).

If we are now to examine a nontrivial X syndrome, an additional step is needed. For example, let us examine a target trajectory of 10$\hat{0}$1 where the first two bits are our X stabilizer measurement outcomes. We now project the results from our last step over these values as below:

$$\left[\begin{array}{c|c|c|c|c|c|c} \hat{z} & II & X_1 & X_2 & X_1X_2 & j & k \\ 00\hat{0}01 & 00\hat{0}01 & -11\hat{1}01 & 00\hat{1}10 & -11\hat{0}10 & 1 & 0 \\ 01\hat{0}00 & 01\hat{0}00 & -10\hat{1}00 & 01\hat{1}11 & -10\hat{0}11 & 1 & 1 \\ 10\hat{1}00 & 10\hat{1}00 & -01\hat{0}00 & 10\hat{0}11 & -01\hat{1}11 & 2 & 1 \\ 11\hat{1}01 & 11\hat{1}01 & -00\hat{0}01 & 11\hat{0}10 & -00\hat{1}10 & 4 & 0 \\ 10\hat{0}11 & 10\hat{0}11 & -01\hat{1}11 & 10\hat{1}00 & -01\hat{0}00 & 3 & 1 \\ 11\hat{0}10 & 11\hat{0}10 & -00\hat{1}10 & 11\hat{1}01 & -00\hat{0}01 & 3 & 0 \\ 00\hat{1}10 & 00\hat{1}10 & -11\hat{0}10 & 00\hat{0}01 & -11\hat{1}01 & 2 & 0 \\ 01\hat{1}11 & 01\hat{1}11 & -10\hat{0}11 & 01\hat{0}00 & -10\hat{1}00 & 4 & 1 \end{array}\right]. \tag{B6}$$

We determine k from the parity of the first d bits (our logical observable, row 5 of Eq. (B1)), and these values can be used to construct the analytical equations (prior to renormalization) for our target. Each term contributes $\alpha^j\beta^{N-j}$ to the corresponding logical state, and we can collect them as below:

$$\begin{pmatrix} |00001\rangle \\ |00110\rangle \\ |11101\rangle \\ |11010\rangle \\ |01000\rangle \\ |01111\rangle \\ |10100\rangle \\ |10011\rangle \end{pmatrix} = \begin{pmatrix} 2\alpha^4\beta + 2\alpha^3\beta^2 - 2\alpha^2\beta^3 - 2\alpha\beta^4 \\ 2\alpha^4\beta + 2\alpha^3\beta^2 - 2\alpha^2\beta^3 - 2\alpha\beta^4 \\ -2\alpha^4\beta - 2\alpha^3\beta^2 + 2\alpha^2\beta^3 + 2\alpha\beta^4 \\ -2\alpha^4\beta - 2\alpha^3\beta^2 + 2\alpha^2\beta^3 + 2\alpha\beta^4 \\ 2\alpha^4\beta - 2\alpha^3\beta^2 - 2\alpha^2\beta^3 + 2\alpha\beta^4 \\ 2\alpha^4\beta - 2\alpha^3\beta^2 - 2\alpha^2\beta^3 + 2\alpha\beta^4 \\ -2\alpha^4\beta + 2\alpha^3\beta^2 + 2\alpha^2\beta^3 - 2\alpha\beta^4 \\ -2\alpha^4\beta + 2\alpha^3\beta^2 + 2\alpha^2\beta^3 - 2\alpha\beta^4 \end{pmatrix}. \tag{B7}$$

Due to the Pauli frame set by our stabilizer measurements, what we consider our logical state is a mix of the following

terms:

$$|\Lambda\rangle = \begin{pmatrix}\alpha_L\\ \beta_L\end{pmatrix} = \begin{pmatrix} |00001\rangle + |00110\rangle - |11101\rangle - |11010\rangle \\ -|01000\rangle - |01111\rangle + |10100\rangle + |10011\rangle \end{pmatrix},$$

$$|\Lambda\rangle = \begin{pmatrix} \alpha^4\beta + \alpha^3\beta^2 - \alpha^2\beta^3 - \alpha\beta^4 \\ -\alpha^4\beta + \alpha^3\beta^2 + \alpha^2\beta^3 - \alpha\beta^4 \end{pmatrix}. \tag{B8}$$

These additional steps for calculating for nontrivial X-stabilizer measurements are costly, and this classical algorithm is only feasible for code distances <6.

APPENDIX C: FUNCTIONAL FORMS FOR $d = 3$ CODE

For $N = 13$ data qubits, there are a total of $2^{(13-1)/2} = 64$ Z-stabilizer trajectories. Each of these trajectories is specified by the 6-bit, Z-parity vector, and each stabilizer trajectory results in a different logical state as a function of $|\chi\rangle^{\otimes N} = (\alpha|0\rangle + \beta|1\rangle)^{\otimes N}$. These functional forms are not normalized.

When initializing a $d = 3$ planar surface code, the specific Pauli frame of the Z stabilizers that are determined after decoding allows you to calculate the actual logically encoded state:

$$000\hat{0}000 = \begin{pmatrix}\alpha^{13} + 4\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 4\alpha^8\beta^5 + 14\alpha^7\beta^6 + 20\alpha^6\beta^7 + 11\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10}\\ 3\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 10\alpha^8\beta^5 + 8\alpha^7\beta^6 + 12\alpha^6\beta^7 + 16\alpha^5\beta^8 + 6\alpha^4\beta^9 + \alpha^2\beta^{11}\end{pmatrix},$$

$$000\hat{0}001 = \begin{pmatrix}\alpha^{12}\beta + \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 5\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11}\\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10}\end{pmatrix},$$

$$000\hat{0}010 = \begin{pmatrix}\alpha^{12}\beta + 2\alpha^{11}\beta^2 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 16\alpha^6\beta^7 + 12\alpha^5\beta^8 + 5\alpha^4\beta^9 + 2\alpha^3\beta^{10}\\ \alpha^{11}\beta^2 + 4\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 8\alpha^4\beta^9 + \alpha^3\beta^{10}\end{pmatrix},$$

$$000\hat{0}011 = \begin{pmatrix}\alpha^{12}\beta + \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 7\alpha^4\beta^9 + \alpha^3\beta^{10}\\ 2\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 6\alpha^4\beta^9 + 4\alpha^3\beta^{10}\end{pmatrix},$$

$$000\hat{1}100 = \begin{pmatrix}\alpha^{12}\beta + \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 5\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11}\\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10}\end{pmatrix},$$

$$000\hat{1}101 = \begin{pmatrix}2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10}\\ 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11}\end{pmatrix},$$

$$000\hat{1}110 = \begin{pmatrix}\alpha^{12}\beta + \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 7\alpha^4\beta^9 + \alpha^3\beta^{10}\\ 2\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 6\alpha^4\beta^9 + 4\alpha^3\beta^{10}\end{pmatrix},$$

$$000\hat{1}111 = \begin{pmatrix}2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 2\alpha^9\beta^4 + 10\alpha^8\beta^5 + 18\alpha^7\beta^6 + 14\alpha^6\beta^7 + 6\alpha^5\beta^8 + 6\alpha^4\beta^9 + 4\alpha^3\beta^{10}\\ 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 14\alpha^8\beta^5 + 14\alpha^7\beta^6 + 6\alpha^6\beta^7 + 10\alpha^5\beta^8 + 10\alpha^4\beta^9 + 2\alpha^3\beta^{10}\end{pmatrix},$$

$$001\hat{0}000 = \begin{pmatrix}\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10}\\ \alpha^{12}\beta + \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 5\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11}\end{pmatrix},$$

$$001\hat{0}001 = \begin{pmatrix}\alpha^{12}\beta + \alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 5\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11}\\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 5\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha\beta^{12}\end{pmatrix},$$

$$001\hat{0}010 = \begin{pmatrix}2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10}\\ \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11}\end{pmatrix},$$

$$001\hat{0}011 = \begin{pmatrix}\alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 2\alpha^9\beta^4 + 10\alpha^8\beta^5 + 20\alpha^7\beta^6 + 12\alpha^6\beta^7 + 6\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11}\\ \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 6\alpha^8\beta^5 + 12\alpha^7\beta^6 + 20\alpha^6\beta^7 + 10\alpha^5\beta^8 + 2\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11}\end{pmatrix},$$

$$001\hat{1}100 = \begin{pmatrix}5\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 4\alpha^8\beta^5 + 14\alpha^7\beta^6 + 18\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11}\\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 18\alpha^7\beta^6 + 14\alpha^6\beta^7 + 4\alpha^5\beta^8 + 6\alpha^4\beta^9 + 5\alpha^3\beta^{10}\end{pmatrix},$$

$$001\hat{1}101 = \begin{pmatrix}\alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 5\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11} + \alpha\beta^{12}\\ 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11}\end{pmatrix},$$

$$001\hat{1}110 = \begin{pmatrix}3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11}\\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10}\end{pmatrix},$$

$$001\hat{1}111 = \begin{pmatrix}\alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11}\\ 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11}\end{pmatrix},$$

$$010\hat{0}000 = \begin{pmatrix}\alpha^{11}\beta^2 + 4\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 8\alpha^4\beta^9 + \alpha^3\beta^{10}\\ \alpha^{12}\beta + 2\alpha^{11}\beta^2 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 16\alpha^6\beta^7 + 12\alpha^5\beta^8 + 5\alpha^4\beta^9 + 2\alpha^3\beta^{10}\end{pmatrix},$$

$$010\hat{0}001 = \begin{pmatrix}2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10}\\ \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11}\end{pmatrix},$$

$$010\hat{0}010 = \begin{pmatrix} \alpha^{12}\beta + 2\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 8\alpha^8\beta^5 + 8\alpha^7\beta^6 + 14\alpha^6\beta^7 + 16\alpha^5\beta^8 + 7\alpha^4\beta^9 \\ \alpha^{11}\beta^2 + 4\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 4\alpha^8\beta^5 + 12\alpha^7\beta^6 + 20\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 3\alpha^3\beta^{10} \end{pmatrix},$$

$$010\hat{0}011 = \begin{pmatrix} \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} \\ \alpha^{11}\beta^2 + 4\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 8\alpha^4\beta^9 + \alpha^3\beta^{10} \end{pmatrix},$$

$$010\hat{0}100 = \begin{pmatrix} 2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix},$$

$$010\hat{0}101 = \begin{pmatrix} 4\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 6\alpha^8\beta^5 + 14\alpha^7\beta^6 + 18\alpha^6\beta^7 + 10\alpha^5\beta^8 + 2\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11} \\ 2\alpha^{10}\beta^3 + 10\alpha^9\beta^4 + 10\alpha^8\beta^5 + 6\alpha^7\beta^6 + 14\alpha^6\beta^7 + 14\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} \end{pmatrix},$$

$$010\hat{0}110 = \begin{pmatrix} \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} \\ \alpha^{11}\beta^2 + 4\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 8\alpha^4\beta^9 + \alpha^3\beta^{10} \end{pmatrix},$$

$$010\hat{0}111 = \begin{pmatrix} 4\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 8\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ 2\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 6\alpha^4\beta^9 + 4\alpha^3\beta^{10} \end{pmatrix},$$

$$011\hat{0}000 = \begin{pmatrix} \alpha^{12}\beta + \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 7\alpha^4\beta^9 + \alpha^3\beta^{10} \\ 2\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 6\alpha^4\beta^9 + 4\alpha^3\beta^{10} \end{pmatrix},$$

$$011\hat{0}001 = \begin{pmatrix} \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 6\alpha^8\beta^5 + 12\alpha^7\beta^6 + 20\alpha^6\beta^7 + 10\alpha^5\beta^8 + 2\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 2\alpha^9\beta^4 + 10\alpha^8\beta^5 + 20\alpha^7\beta^6 + 12\alpha^6\beta^7 + 6\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix},$$

$$011\hat{0}010 = \begin{pmatrix} \alpha^{11}\beta^2 + 4\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 8\alpha^4\beta^9 + \alpha^3\beta^{10} \\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} \end{pmatrix},$$

$$011\hat{0}011 = \begin{pmatrix} 2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11} \end{pmatrix},$$

$$011\hat{0}100 = \begin{pmatrix} \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} \\ 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix},$$

$$011\hat{0}101 = \begin{pmatrix} 4\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 8\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ \alpha^{10}\beta^3 + 7\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} + \alpha\beta^{12} \end{pmatrix},$$

$$011\hat{0}110 = \begin{pmatrix} \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 10\alpha^8\beta^5 + 6\alpha^7\beta^6 + 14\alpha^6\beta^7 + 16\alpha^5\beta^8 + 6\alpha^4\beta^9 + \alpha^3\beta^{10} \\ \alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 16\alpha^8\beta^5 + 14\alpha^7\beta^6 + 6\alpha^6\beta^7 + 10\alpha^5\beta^8 + 8\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix},$$

$$011\hat{0}111 = \begin{pmatrix} 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 4\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix},$$

$$100\hat{0}000 = \begin{pmatrix} \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} \\ \alpha^{12}\beta + \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 5\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix},$$

$$100\hat{0}001 = \begin{pmatrix} 5\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 4\alpha^8\beta^5 + 14\alpha^7\beta^6 + 18\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 18\alpha^7\beta^6 + 14\alpha^6\beta^7 + 4\alpha^5\beta^8 + 6\alpha^4\beta^9 + 5\alpha^3\beta^{10} \end{pmatrix},$$

$$100\hat{0}010 = \begin{pmatrix} 2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix},$$

$$100\hat{0}011 = \begin{pmatrix} 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} \end{pmatrix},$$

$$100\hat{0}100 = \begin{pmatrix} \alpha^{12}\beta + \alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 5\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 5\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha\beta^{12} \end{pmatrix},$$

$$100\hat{0}101 = \begin{pmatrix} \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 5\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11} + \alpha\beta^{12} \\ 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix},$$

$$100\hat{0}110 = \begin{pmatrix} \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 2\alpha^9\beta^4 + 10\alpha^8\beta^5 + 20\alpha^7\beta^6 + 12\alpha^6\beta^7 + 6\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 6\alpha^8\beta^5 + 12\alpha^7\beta^6 + 20\alpha^6\beta^7 + 10\alpha^5\beta^8 + 2\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix},$$

$$100\hat{0}111 = \begin{pmatrix} \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11} \end{pmatrix},$$

$$101\hat{0}000 = \begin{pmatrix} 2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11} \end{pmatrix},$$

$$
\begin{aligned}
101\hat{1}001 &= \begin{pmatrix} 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 5\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11} + \alpha\beta^{12} \end{pmatrix}, \\
101\hat{1}010 &= \begin{pmatrix} 2\alpha^{10}\beta^3 + 10\alpha^9\beta^4 + 10\alpha^8\beta^5 + 6\alpha^7\beta^6 + 14\alpha^6\beta^7 + 14\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ 4\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 6\alpha^8\beta^5 + 14\alpha^7\beta^6 + 18\alpha^6\beta^7 + 10\alpha^5\beta^8 + 2\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11} \end{pmatrix}, \\
101\hat{1}011 &= \begin{pmatrix} 4\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 8\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ \alpha^{10}\beta^3 + 7\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} + \alpha\beta^{12} \end{pmatrix}, \\
101\hat{1}100 &= \begin{pmatrix} 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 5\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + \alpha^3\beta^{10} + \alpha^2\beta^{11} + \alpha\beta^{12} \end{pmatrix}, \\
101\hat{1}101 &= \begin{pmatrix} \alpha^{11}\beta^2 + 6\alpha^9\beta^4 + 16\alpha^8\beta^5 + 12\alpha^7\beta^6 + 8\alpha^6\beta^7 + 10\alpha^5\beta^8 + 8\alpha^4\beta^9 + 3\alpha^3\beta^{10} \\ 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 11\alpha^8\beta^5 + 20\alpha^7\beta^6 + 14\alpha^6\beta^7 + 4\alpha^5\beta^8 + 4\alpha^4\beta^9 + 4\alpha^3\beta^{10} + \beta^{13} \end{pmatrix}, \\
101\hat{1}110 &= \begin{pmatrix} 4\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 8\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ \alpha^{10}\beta^3 + 7\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} + \alpha\beta^{12} \end{pmatrix}, \\
101\hat{1}111 &= \begin{pmatrix} \alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 4\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ 2\alpha^{10}\beta^3 + 5\alpha^9\beta^4 + 12\alpha^8\beta^5 + 16\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^2\beta^{11} + \alpha\beta^{12} \end{pmatrix}, \\
110\hat{0}000 &= \begin{pmatrix} \alpha^{12}\beta + \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 7\alpha^4\beta^9 + \alpha^3\beta^{10} \\ 2\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 6\alpha^4\beta^9 + 4\alpha^3\beta^{10} \end{pmatrix}, \\
110\hat{0}001 &= \begin{pmatrix} \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} \\ 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix}, \\
110\hat{0}010 &= \begin{pmatrix} \alpha^{11}\beta^2 + 4\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 12\alpha^5\beta^8 + 8\alpha^4\beta^9 + \alpha^3\beta^{10} \\ \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} \end{pmatrix}, \\
110\hat{0}011 &= \begin{pmatrix} \alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 10\alpha^8\beta^5 + 6\alpha^7\beta^6 + 14\alpha^6\beta^7 + 16\alpha^5\beta^8 + 6\alpha^4\beta^9 + \alpha^3\beta^{10} \\ \alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 16\alpha^8\beta^5 + 14\alpha^7\beta^6 + 6\alpha^6\beta^7 + 10\alpha^5\beta^8 + 8\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix}, \\
110\hat{0}100 &= \begin{pmatrix} \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 6\alpha^8\beta^5 + 12\alpha^7\beta^6 + 20\alpha^6\beta^7 + 10\alpha^5\beta^8 + 2\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{11}\beta^2 + 3\alpha^{10}\beta^3 + 2\alpha^9\beta^4 + 10\alpha^8\beta^5 + 20\alpha^7\beta^6 + 12\alpha^6\beta^7 + 6\alpha^5\beta^8 + 6\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix}, \\
110\hat{0}101 &= \begin{pmatrix} 4\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 8\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ \alpha^{10}\beta^3 + 7\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} + \alpha\beta^{12} \end{pmatrix}, \\
110\hat{0}110 &= \begin{pmatrix} 2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 12\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11} \end{pmatrix}, \\
110\hat{0}111 &= \begin{pmatrix} 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ \alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 4\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix}, \\
111\hat{0}000 &= \begin{pmatrix} 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 14\alpha^8\beta^5 + 14\alpha^7\beta^6 + 6\alpha^6\beta^7 + 10\alpha^5\beta^8 + 10\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ 2\alpha^{11}\beta^2 + 2\alpha^{10}\beta^3 + 2\alpha^9\beta^4 + 10\alpha^8\beta^5 + 18\alpha^7\beta^6 + 14\alpha^6\beta^7 + 6\alpha^5\beta^8 + 6\alpha^4\beta^9 + 4\alpha^3\beta^{10} \end{pmatrix}, \\
111\hat{0}001 &= \begin{pmatrix} \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11} \end{pmatrix}, \\
111\hat{0}010 &= \begin{pmatrix} 4\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 8\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 8\alpha^4\beta^9 + 2\alpha^3\beta^{10} \\ 2\alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 10\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 6\alpha^4\beta^9 + 4\alpha^3\beta^{10} \end{pmatrix}, \\
111\hat{0}011 &= \begin{pmatrix} \alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 4\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix}, \\
111\hat{0}100 &= \begin{pmatrix} \alpha^{11}\beta^2 + \alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 3\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ 2\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 12\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^3\beta^{10} + 2\alpha^2\beta^{11} \end{pmatrix}, \\
111\hat{0}101 &= \begin{pmatrix} 2\alpha^{10}\beta^3 + 5\alpha^9\beta^4 + 12\alpha^8\beta^5 + 16\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 4\alpha^4\beta^9 + 2\alpha^2\beta^{11} + \alpha\beta^{12} \\ \alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 4\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix}, \\
111\hat{0}110 &= \begin{pmatrix} \alpha^{10}\beta^3 + 8\alpha^9\beta^4 + 12\alpha^8\beta^5 + 12\alpha^7\beta^6 + 14\alpha^6\beta^7 + 8\alpha^5\beta^8 + 4\alpha^4\beta^9 + 4\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ 3\alpha^{10}\beta^3 + 6\alpha^9\beta^4 + 10\alpha^8\beta^5 + 14\alpha^7\beta^6 + 12\alpha^6\beta^7 + 10\alpha^5\beta^8 + 6\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha^2\beta^{11} \end{pmatrix}, \\
111\hat{0}111 &= \begin{pmatrix} 3\alpha^{10}\beta^3 + 4\alpha^9\beta^4 + 10\alpha^8\beta^5 + 20\alpha^7\beta^6 + 12\alpha^6\beta^7 + 4\alpha^5\beta^8 + 6\alpha^4\beta^9 + 4\alpha^3\beta^{10} + \alpha^2\beta^{11} \\ 7\alpha^9\beta^4 + 16\alpha^8\beta^5 + 14\alpha^7\beta^6 + 8\alpha^6\beta^7 + 8\alpha^5\beta^8 + 8\alpha^4\beta^9 + 2\alpha^3\beta^{10} + \alpha\beta^{12} \end{pmatrix}.
\end{aligned} \tag{C1}
$$

APPENDIX D: SPECIFIC STATES PRODUCED VIA TRANSVERSAL INJECTION

In this section, we give an example of a specific solution for each of the expressions in Eq. (C1), for the $d = 3$ codes where an $|A\rangle$ state, suitable to implement the T-gate exists, for an initial state, written in polar form, $(|\chi\rangle)^{\otimes 13} = [\cos(\frac{\theta}{2})|0\rangle + e^{i\phi}\sin(\frac{\theta}{2})|1\rangle]^{\otimes 13}$, where $(\theta = 2.44580563149781, \phi = 1.3616970885685595)$ (Table II). It should be noted that this solution is not unique.

The trivial X and Z trajectories corresponding to the binary vectors $010\hat{0}011$ and $010\hat{1}110$ produce the state $(|0\rangle_L + e^{-i\pi/4}|1\rangle_L)/\sqrt{2}$, while the Z trajectories corresponding to the binary vectors $011\hat{0}10$ and $110\hat{0}10$ produce the state $(|0\rangle_L + e^{i\pi/4}|1\rangle_L)/\sqrt{2}$. Both of these states can be used to realize a logical T-gate. All 64 states produced for a $d = 3$ code with the input above are illustrated in Fig. 8.

It is the focus of future work to investigate the states produced by transversal injection and how these states can be utilized even though their preparation is probabilistic but heralded.

[1] S. J. Devitt, W. J. Munro, and K. Nemoto, Quantum error correction for beginners, Rep. Prog. Phys. **76**, 076001 (2013).

[2] B. M. Terhal, Quantum error correction for quantum memories, Rev. Mod. Phys. **87**, 307 (2015).

[3] T. Rudolph, Why I am optimistic about the silicon-photonic route to quantum computing, APL Photonics **2**, 030901 (2017).

[4] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[5] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, Elucidating reaction mechanisms on quantum computers, Proc. Natl. Acad. Sci. USA **114**, 7555 (2017).

[6] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity, Phys. Rev. X **8**, 041015 (2018).

[7] V. von Burg, G. H. Low, T. Häner, D. S. Steiger, M. Reiher, M. Roetteler, and M. Troyer, Quantum computing enhanced computational catalysis, Phys. Rev. Res. **3**, 033055 (2021).

[8] D. A. Lidar and T. A. Brun, *Quantum Error Correction* (Cambridge University Press, Cambridge, 2013).

[9] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[10] B. Lekitsch, S. Weidt, A. G. Fowler, K. Mølmer, S. J. Devitt, C. Wunderlich, and W. K. Hensinger, Blueprint for a microwave trapped ion quantum computer, Sci. Adv. **3**, e1601540 (2017).

[11] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, Layered Architecture for Quantum Computing, Phys. Rev. X **2**, 031007 (2012).

[12] H. Mukai, K. Sakata, S. J. Devitt, R. Wang, Y. Zhou, Y. Nakajima, and J.-S. Tsai, Pseudo-2D superconducting quantum computing circuit for the surface code: Proposal and preliminary tests, New J. Phys. **22**, 043013 (2020).

[13] C. D. Hill, E. Peretz, S. J. Hile, M. G. House, M. Fuechsle, S. Rogge, M. Y. Simmons, and L. C. L. Hollenberg, A surface code quantum computer in silicon, Sci. Adv. **1**, e1500707 (2015).

[14] H. Bombin, I. H. Kim, D. Litinski, N. Nickerson, M. Pant, F. Pastawski, S. Roberts, and T. Rudolph, Interleaving: Modular architectures for fault-tolerant photonic quantum computing, arXiv:2103.08612.

[15] C. Gidney and M. Ekerå, How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, Quantum **5**, 433 (2021).

[16] O. D. Matteo, V. Gheorghiu, and M. Mosca, Fault-tolerant resource estimation of quantum random-access memories, IEEE Trans. Quantum Eng. **1**, 1 (2020).

[17] D. Litinski, A game of surface codes: Large-scale quantum computing with lattice surgery, Quantum **3**, 128 (2019).

[18] A. G. Fowler and C. Gidney, Low overhead quantum computation using lattice surgery, arXiv:1808.06709.

[19] C. Gidney and A. G. Fowler, Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation, Quantum **3**, 135 (2019).

[20] S. J. Devitt, A. M. Stephens, W. J. Munro, and K. Nemoto, Requirements for fault-tolerant factoring on an atom-optics quantum computer, Nat. Commun. **4**, 2524 (2013).

[21] A. M. Stephens, Fault-tolerant thresholds for quantum error correction with the surface code, Phys. Rev. A **89**, 022321 (2014).

[22] Multiple studies have numerically derived the precise scaling, including constant factors for a required logical error rate in the planar surface code as a function of additional hardware constraints, more precise physical error models, and overall physical error rate [21].

[23] B. Q. Baragiola, G. Pantaleoni, R. N. Alexander, A. Karanjai, and N. C. Menicucci, All-Gaussian Universality and Fault Tolerance with the Gottesman-Kitaev-Preskill Code, Phys. Rev. Lett. **123**, 200502 (2019).

[24] C. Horsman, A. G. Fowler, S. Devitt, and R. van Meter, Surface code quantum computing by lattice surgery, New J. Phys. **14**, 123011 (2012).

[25] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, Phys. Rev. A **71**, 022316 (2005).

[26] D. Litinski, Magic state distillation: Not as costly as you think, Quantum **3**, 205 (2019).

[27] B. Eastin and E. Knill, Restrictions on Transversal Encoded Quantum Gate Sets, Phys. Rev. Lett. **102**, 110502 (2009).

[28] H. Bombín, Gauge color codes: Optimal transversal gates and gauge fixing in topological stabilizer codes, New J. Phys. **17**, 083002 (2015).

[29] S. Bravyi and J. Haah, Magic-state distillation with low overhead, Phys. Rev. A **86**, 052329 (2012).

[30] Y. Li, A magic state's fidelity can be superior to the operations that created it, New J. Phys. **17**, 023037 (2015).

[31] S. Aaronson and D. Gottesman, Improved simulation of stabilizer circuits, Phys. Rev. A **70**, 052328 (2004).

[32] J. R. Johansson, P. D. Nation, and F. Nori, QuTiP 2: A Python framework for the dynamics of open quantum systems, Comput. Phys. Commun. **184**, 1234 (2013).

[33] C. Gidney, Stim: A fast stabilizer circuit simulator, Quantum **5**, 497 (2021).