Sdim: A Qudit Stabilizer Simulator

Adeeb Kabir*
Steven Nguyen*
Tijil Kiran
Yipeng Huang
ask171@rutgers.edu
snn28@scarletmail.rutgers.edu
tk664@scarletmail.rutgers.edu
yipeng.huang@rutgers.edu
Rutgers University
Piscataway, New Jersey, USA

Sohan Ghosh
Isaac H. Kim
sohghosh@ucdavis.edu
ikekim@ucdavis.edu
University of California, Davis
Davis, California, USA

Abstract

Quantum computers have steadily improved over the last decade, but developing fault-tolerant quantum computing (FTQC) techniques, required for useful, universal computation remains an ongoing effort. Key elements of FTQC such as error-correcting codes and decoding are supported by a rich bed of stabilizer simulation software such as Stim and CHP, which are essential for numerically characterizing these protocols at realistic scales. Recently, experimental groups have built nascent high-dimensional quantum hardware, known as qudits, which have a myriad of attractive properties for algorithms and FTQC. Despite this, there are no widely available qudit stabilizer simulators. We introduce the first open-source realization of such a simulator for all dimensions. We demonstrate its correctness against existing state vector simulations and benchmark its performance in evaluating and sampling quantum circuits. This simulator is the essential computational infrastructure to explore novel qudit error correction as earlier stabilizer simulators have been for qubits.

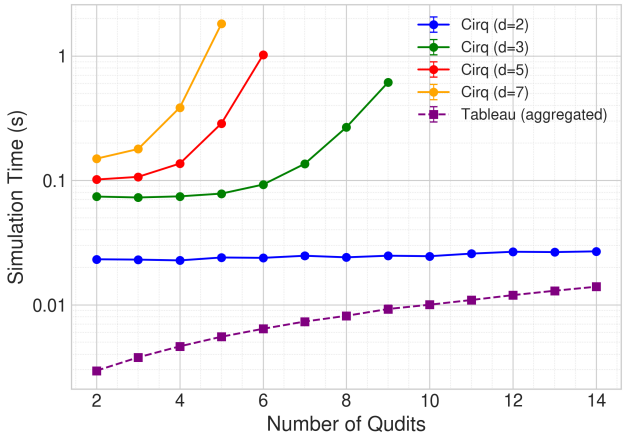


Figure 1. Average time to simulate one shot of a random Clifford circuit using our stabilizer versus Google Cirq statevector simulation. The space and time cost of state vector simulation grows exponentially versus the dimension and number of qudits. In contrast, tableau simulation does not depend on the dimension at all and is bounded by n^2.

1 Introduction

Despite the remarkable progress in the construction of noisy intermediate-scale quantum (NISQ) systems and the identification of NISQ applications [3, 20, 37, 58, 60], the future of computing is the construction of fault-tolerant quantum computing (FTQC) systems [9, 24, 55]. The quantum error correction code (QECC), which produces a noise-resistant logical qubit out of many noisy physical qubits, is an object of central study in the FTQC paradigm [21, 33, 57].

The conventional wisdom is that QECCs in FTQC systems will be focused narrowly on stabilizer, topological, surface, and toric codes in qubit quantum systems [19, 39, 40, 68]. The two-dimensional topology of surface codes is well-suited to the two-dimensional locally connected topologies in prototype superconducting quantum computers. These codes natively support a subset of operations termed Clifford operators, and any non-Clifford operations are introduced via magic state distillation and injection [22]. However, recent work suggests that at least 100 physical qubits are needed to realize a single logical qubit [2]-and maintaining error rates below threshold at that scale is difficult. The need for the magic-state distillation protocol adds additional overheads to any FTQC algorithm execution.

Two trends in quantum device engineering motivate the consideration of novel QECC protocols. First, emerging quantum devices allow for more dense connections between qubits and further allow for grouped movement of multiple qubits at once. These abilities enable QECCs where qubit connectivity is less local and is not restricted to a two-dimensional plane [69]. Second, most quantum devices support quantum states beyond the lowest two typically used to encode qubits. These states, termed *qudits*, were once considered much more unreliable to access and control than qubits. Recent developments have shown remarkable progress in controlling these states [10, 29, 47, 48, 63, 70], and these qudit states have already had an impact on NISQ

*Authors contributed equally to this research.

systems [4, 5, 10, 18, 23, 26, 28, 46, 47, 53, 61, 64, 67]. The question is what their impact will be on FTQC systems.

While all QECC research begins with the analytical derivation of codes, the computational implementation of the codes is vital to check their correctness and to numerically characterize critical properties such as their physical noise threshold. Simulations based on the restricted set of Pauli-stabilized quantum states and Clifford gate operations [1, 27, 30] have been critical for qubit QECC research because, at the scale of physical qubits involved in QECCs, conventional numerical simulation based on matrix vector multiplication on state vectors is prohibitively expensive, seen in Figure 1.

On the other hand, such work is absent in qudit QECC research as all available software support for qudit simulation exists only in the non-scalable state vector style [11, 16, 44, 52], despite the theory of qudit stabilizer computation having long since been well understood [15, 32].

This work introduces an *open-source qudit stabilizer simulator* as the missing piece towards the study of qudit QECCs. This simulator is *tableau*-based, where the state of n d-dimensional qudits is compactly represented in a table of $O(n^2)$ integers compared to the $O(d^n)$ floating point numbers needed for state vectors. Circuit evaluation becomes a sequence of simple update rules applied on the table for each operator in the circuit.

Furthermore, this work contributes an efficient Monte Carlo sampler for measurements, both with and without noise trajectories. This is achieved via qudit *Pauli frames*, first used in [27] to efficiently sample qubit measurements. Altogether, Sdim is the minimum needed to efficiently simulate large-scale, practical qudit stabilizer error correction circuits on classical hardware. The sampler in particular was crafted to support numerical characterization of a 5-qutrit chip protected by a *folded surface code* [56]. This protocol outputs a *logical error rate* of the protected system against a single physical qutrit's error rate.

This simulator is functional for all dimensions $d > 2$, but the non-prime dimensions involve technical challenges that build off the prime case, and so we will restrict our attention exclusively to high-dimensional prime dimensions for the majority of this paper.

In short, we offer the following contributions:

- We introduce Sdim, a qudit stabilizer simulator functional for all $d \geq 2$, and describe its components and API for any prime dimension d.
- We validate Sdim's correctness and performance against Google Cirq, the most prominent high-dimensional qudit state vector simulator.
- We introduce qudit Pauli frames, which are used to efficiently sample qudit circuits, and we validate its correctness and performance against Cirq's sampling.
- We demonstrate the Sdim's immediate utility by validating a Cirq-simulated logical randomized benchmarking error characterization of 5-qutrit hardware protected by a novel error *detecting* code. Our simulator exhibits significantly greater performance even in this relatively small state space.
- We briefly discuss challenges and differences present for non-prime dimension d.

2 Motivation for characterizing qudit QECC

The motivation for computational validation and numerical characterization of qudit QECCs is two-fold.

First, recent experiments in quantum devices are changing the assumptions that have motivated the dominant form of QECCs so far. The majority of research efforts on the QECCs underlying FTQC systems have been focused on qubit stabilizer surface codes that support Clifford operations within the code [19], while non-Clifford operations must be injected through a separate protocol. However, the steady development of accessing and controlling qudit states [7, 10, 12, 23, 26, 29, 38, 45, 47, 48, 61–64, 66, 70] raises the possibility that the first demonstration of FTQC might take place in a qudit system at relatively high physical error rates, rather than in a qubit system at necessarily lower physical error rates [8].

Second, the types of QECC that researchers can fully study (analytically derive, computationally validate, numerically characterize) for now are limited by what classical computers can simulate [27, 41]. As a result, nearly all QECCs for which there are numerical physical error threshold studies belong to qubit stabilizer codes. Qudit simulation has received attention from industrial and academic research in the past [11, 16, 44, 52], but all such simulators are based on state vector simulation, and not the scalable variety for stabilizer circuits.

Conventional wisdom posits that qudit codes exhibit higher rates than qubit codes and that they can generally correct more errors [33], but the lack of efficient simulators have left these claims largely uncontested. Indeed, despite recent attention provoking further interrogation [41], these studies are limited by the (in)feasibility of state vector qudit simulation. Any nonstabilizer code would be computationally too costly to simulate. Given that stabilizer simulation is what is scalable, it is possible to numerically characterize qudit stabilizer codes. This paper is the first open-source implementation of such a tool.

3 Background on qudit stabilizers

This section, reviews the essentials of quantum stabilizer computation, particularly the representation of quantum states as a table of integers known as a *tableau*. As an aside, this formalism is exactly the same as describing a stabilizer code [31, 33].

3.1 Qudit Pauli stabilizer states & Clifford operations

States. A dimension d qudit is a quantum system with d computational basis states $\{|0\rangle, \ldots, |d-1\rangle\}$. We write product states in various ways: $|k_1 k_2 \ldots k_n\rangle = |k_1\rangle \otimes |k_2\rangle \otimes \cdots \otimes |k_n\rangle = |k_1\rangle |k_2\rangle \ldots |k_n\rangle$. All qudit states are either product states or sums of product states.

Gates. A single qudit gate G acting locally on the qudit l is written G_l, and when gates $G_1, \ldots, G_m$ act locally on distinct qudits in state $|k_1 k_2 \ldots k_n\rangle$, we similarly omit the product symbol for brevity: $G_1 G_2 \ldots G_m$. If $m < n$, then at least one G_l acts on the $q > 1$ qudits, which, as written, means that it acts locally on the product state $|k_l k_{l+1} \ldots k_{l+q}\rangle$.

Pauli operators. The *Pauli group* $\mathcal{P}$ is ubiquitous in quantum computing as both a set of elementary operations and a discrete basis for arbitrary errors [17, 54, 57, 59]. The qudit Pauli group $\mathcal{P}(d)$ is generated by gates X and Z (identified as *flip* and *phase* errors) where $X|j\rangle = |(j+1) \mod d\rangle$ and $Z|j\rangle = \omega^j |j\rangle$, where the root of unity $\omega = \exp(2\pi i/d)$. We note the anti-commutation relation $ZX = \omega XZ$. All Pauli operators, including the "Y"-type operators can simply be written as a product of X and Z up to phase, so we always express and identify the elements of $\mathcal{P}$ in that form. Explicitly, we may write *any* of them as $\omega^c X^a Z^b$, identified by a 3-tuple integer encoding (mod d) (a, b, c). Tensor products of Pauli operators are commonly called *Pauli strings*.

Clifford operators. The qudit *Clifford group* $C := C(d)$ consists of all gates C that *normalize* the Pauli group, meaning $CWC^\dagger \in \mathcal{P}$, for any $W \in \mathcal{P}$. The qubit Cliffords are commonly written as gates generated by $\{CNOT, H, P\}$ [1, 30], each of which is typically generalized as $\{SUM, \mathcal{F}, P'\}$ for $d > 2$. These gates act as follows:

$$SUM(|i\rangle \otimes |j\rangle) = |i\rangle |(i+j) \mod d\rangle$$

$$\mathcal{F}|i\rangle = \sum_{j=0}^{d-1} \omega^{ij} |j\rangle$$

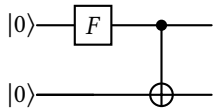
$$P'|j\rangle = \omega^{j(j-1)/2} |j\rangle$$

The SUM gate conditionally applies X^i on the target wire, and $\mathcal{F}$ is the d-dimensional discrete Fourier transform. The phase gate P' appends a phase that is quadratic in its computational basis label [33].

As an example, consider the state

$$|\phi\rangle = \frac{|00\rangle + |11\rangle + |22\rangle}{\sqrt{3}} \tag{1}$$

which is the result of the following Clifford circuit:



Clifford operations alone cannot achieve universal quantum computation. In fact, a choice of any single non-Clifford operation completes the set of gates needed for universal QC. A common choice is the T gate, creating the Clifford+T universal gate set. However, simulating this gate set is costly in general [9].

$\mathcal{F}$	P'
$X \to Z$	$X \to XZ$
$Z \to X^{-1}$	$Z \to Z$
SUM	
$X \otimes I \to X \otimes X$	$I \otimes X \to I \otimes X$
$Z \otimes I \to Z \otimes I$	$I \otimes Z \to Z^{-1} \otimes Z$

Figure 2. Conjugation table for Clifford gates on Pauli gates.

3.2 Classical simulation of qudit stabilizer circuits

Although Clifford operations on Pauli-stabilized states are not enough to describe universal QC, they are expressive enough to describe and simulate the broad class of states that can be protected by (stabilizer) codes.

Stabilizers of a quantum state. Simply put, a *stabilizer* G of $|\psi\rangle$ is a Pauli gate G such that $G|\psi\rangle = |\psi\rangle$. The set of these gates forms a *unique* group $S := S(|\psi\rangle)$ under multiplication, called a *stabilizer group*. Returning to example from Equation 1, we may readily verify that the following operators stabilize $|\phi\rangle$ and are thus elements of $\mathcal{S}(|\phi\rangle)$:

$$I_0 I_1,\ X_0^2 X_1^2,\ Z_0^2 Z_1,\ \ldots,\ X_0^2 Z_0^2 X_1^2 Z_1 \tag{2}$$

Stabilizer states. An *n-fold stabilizer state* $|\psi\rangle$ is the result of a Clifford circuit applied to $|0\rangle^{\otimes n}$ (also called a *stabilizer circuit*). Clearly, our favorite example $|\phi\rangle$ is a stabilizer state. **A basic but important fact** about $|\psi\rangle$ is that *any* length n list of its non-identity, independent stabilizers $g_1, g_2, \ldots, g_n \in S$ *generates* S. For $|\phi\rangle$, every single one of the stabilizers in equation 2 can be written as products of $X_0^2 X_1^2$ and $Z_0^2 Z_1$. Furthermore, $X_0^2 X_1^2$ cannot be written as a power of $Z_0^2 Z_1$ and vice versa, so they are independent. In this way, these two operators generate $\mathcal{S}(|\phi\rangle)$.

Another example is $|0\rangle^{\otimes n}$, which is stabilized by operators $Z_1, Z_2, \ldots, Z_n$. These are n distinct stabilizers, so their products describe other stabilizers of $|0\rangle^{\otimes n}$ such as $Z_1 Z_2$. Naturally, this means that the stabilizer generators are a succinct representation of $|\psi\rangle$ [30].

Stabilizer state evolution. Now, suppose that C is a Clifford gate and that we want to describe $|\psi\rangle = C|0\rangle^{\otimes n}$ via stabilizer group generators. Notice:

$$|\psi\rangle = C|0\rangle^{\otimes n} = CZ_j|0\rangle^{\otimes n} = (CZ_jC^\dagger)C|0\rangle^{\otimes n} = (CZ_jC^\dagger)|\psi\rangle$$

Operator $CZ_jC^\dagger$ is Pauli by definition, and from the above equation, the operators $\{CZ_1C^\dagger, \ldots, CZ_nC^\dagger\}$ stabilize $|\psi\rangle$, so they are valid stabilizer generators for $|\psi\rangle$. *Clifford evolution is a game of conjugating stabilizer generators.*

```
1  is_constant = random.choice(["True", "False"])
2  c = Circuit(2, d)  # Initial circuit is |00>
3  c.add_gate("H", 0) # F |0> on qudit 0
4  c.add_gate("X", 1) # Flip ancilla |0> to |1>
5  c.add_gate("H", 1) # F |1> on ancilla
6  if is_constant:
7      function_constant = random.choice([j for j in
       range(d)])
8      for i in range(function_constant):
9          c.add_gate("X", 1) # Oracle for f=j
10 else:
11     c.add_gate("CNOT", 0, 1) # Oracle f=id
12 c.add_gate("H_INV", 0) # F^t |0> on qudit 0
13 c.add_gate("M", 0) # Measure qudit 0
14 expected = 0 if is_constant else d - 1
15 p = Program(c)
16 assert p.simulate() == [MeasurementResult(0, True,
       expected)]
```

Figure 3. A Sdim program to run and validate the d-dimensional Deutsch-Jozsa algorithm.

Stabilizer integer encoding. Recall that every Pauli $P = \omega^c X^a Z^b$. Every product Pauli $P_1 \otimes \cdots \otimes P_m$ can then be written $\omega^r (X^{a_1} Z^{b_1} \otimes \cdots \otimes X^{a_1} Z^{b_1})$, where $r = c_1 + \cdots + c_m$. We can write it out in a compact *block form* (sometimes known as its *symplectic form* [33]):

$$\left[\begin{array}{ccc|ccc|c} a_1 & \ldots & a_m & b_1 & \ldots & b_m & r \end{array}\right]$$

The two leftmost blocks are the X-block and Z-*block*, respectively, followed by the *phase block*. The product of two stabilizers in the symplectic form is simply the entry-wise sum of their respective symplectic forms. The conjugation of a stabilizer by a Clifford operator amounts to simple rewrite rules across the blocks, which are derived from the table in figure 2.

4 Validation and evaluation of qudit stabilizer tableau simulation

Sdim is best understood in action. Throughout this section, we will introduce its key components and features. As a guided example, we will demonstrate how these components arise in a simple quantum circuit. The tutorial case here will be a circuit that runs a qutrit variant of the *Deutsch-Jozsa algorithm.*

4.1 Implementation of our open source qudit stabilizer tableau simulator

Sdim is a Python library that that creates and evaluates quantum circuits. The program in Figure 3 shows how to write a circuit with most of the major user features and how to read out a measurement.

Data primitives. The basic data the user manipulates is a `Circuit` object. To create an an empty n-qudit circuit of dimension d, we invoke the constructor `c = Circuit(n, d)`. The object is initialized to the state $|0\rangle^{\otimes n}$. Once the user has written out a circuit using the operations detailed below, they can simulate it by creating a `Program` object constructed out of a circuit. This is seen in line 15 of Figure 3.

Gate operations and measurements. The user appends gate instructions to the circuit via its `add_gate` function, which takes arguments for a gate name and a local qudit index (and additional target for `CNOT` when applicable), seen in lines 3-5, 11 of Figure 3 . The elementary gate set is `{X, Z, SUM, F, P}` along with their inverse gates `{X_INV, Z_INV, SUM_INV, F_INV, P_INV}`. The gates can be called by either their qubit or qudit names (e.g. `H` versus `F` on line 3 of Figure 3) with no affect on the program. At any time, the user can reset the j^{th} physical qudit to $|0\rangle$ using the `RESET` gate on index j.

"Measurement at an index j" is shorthand for performing a measurement of operator Z_j, which collapses (at least) the j^{th} physical qudit. It is appended to the circuit in the same way as a Clifford gate, seen on line 14 of Figure 3.

Error channels. Our simulator has a gate `N1` that implements common single-qudit noise channels. Given a positional parameter j, a *channel* parameter `t`, and a probability `p`, the code:

```
c.add_gate('N1', noise_channel=t, prob=p)
```

applies noise channel of type `t` to qudit j with probability `p`, where the types are `'f'`, `'p'`, `'d'` corresponding to *flip*, *phase*, and *depolarizing* noise, respectively.

Simulation. Finally, the user may invoke the `simulate` method on a `Program` object to evaluate the circuit. There is an optional `shots` parameter that allows one to sample the circuit many times, which we will return to later. The simulation produces a 3D list of `MeasurementResult` objects, indexed by shot number, the index of the measurement, and sequential order of the measurement in the circuit. The `MeasurementResult` itself stores the qudit index, the measurement outcome, and whether it was deterministic. If only a single shot is simulated, the list is 2D. Line 16 of Figure 3 demonstrates checking whether a single terminal measurement of Deutsch-Jozsa on the first qudit is either 0 or $d-1$.

4.2 Tutorial case study of Qutrit Deutsch-Jozsa algorithm simulation

The Deutsch-Jozsa problem is as follows: Given sets $A = \{0, \ldots, d-1\}^n$ and $B = \{0, \ldots, d-1\}$, suppose an unknown map $f : A \to B$ is either *constant* or *balanced*, where being

Source code	`c = Circuit(2, 3)`	`c.add_gate("H", 0)`	`c.add_gate("X", 1)`	`c.add_gate("H", 1)`
Gate operation	Initialization	$\mathcal{F} \otimes I$	$I \otimes X$	$I \otimes \mathcal{F}$
Gate unitary	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes I$	$\frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix} \otimes I$	$I \otimes \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$I \otimes \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix}$
Statevector	$\lvert 0\rangle \otimes \lvert 0\rangle$	$\lvert +\rangle \otimes \lvert 0\rangle$	$\lvert +\rangle \otimes \lvert 1\rangle$	$\lvert +\rangle \otimes \lvert \omega\rangle$
Stabilizers	$\begin{matrix} II & IZ & IZ^2 \\ ZI & ZZ & ZZ^2 \\ Z^2I & Z^2Z & Z^2Z^2 \end{matrix}$	$\begin{matrix} II & IZ & IZ^2 \\ X^2I & X^2Z & X^2Z^2 \\ XI & XZ & XZ^2 \end{matrix}$	$\begin{matrix} II & I\omega^2Z & I\omega Z^2 \\ X^2I & X^2\omega^2Z & X^2\omega Z^2 \\ XI & X\omega^2Z & X\omega Z^2 \end{matrix}$	$\begin{matrix} II & I\omega^2X^2 & I\omega X \\ X^2I & X^2\omega^2X^2 & X^2\omega X \\ XI & X\omega^2X^2 & X\omega X \end{matrix}$
Destabilizer generators	$\{XI, IX\}$	$\{ZI, IX\}$	$\{ZI, IX\}$	$\{ZI, IZ\}$
Stabilizer generators	$\{ZI, IZ\}$	$\{X^2I, IZ\}$	$\{X^2I, \omega^2 IZ\}$	$\{X^2I, \omega^2 IX^2\}$
Tableau	$\left[\begin{array}{cc\|cc\|c} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$	$\left[\begin{array}{cc\|cc\|c} \underline{0} & 0 & \underline{1} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline \underline{2} & 0 & \underline{0} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$	$\left[\begin{array}{cc\|cc\|c} 0 & 0 & 1 & 0 & 0 \\ 0 & \underline{1} & 0 & \underline{0} & 0 \\ \hline 2 & 0 & 0 & 0 & 0 \\ 0 & \underline{0} & 0 & \underline{1} & \underline{2} \end{array}\right]$	$\left[\begin{array}{cc\|cc\|c} 0 & 0 & 1 & 0 & 0 \\ 0 & \underline{0} & 0 & \underline{1} & 0 \\ \hline 2 & 0 & 0 & 0 & 0 \\ 0 & \underline{2} & 0 & \underline{0} & 2 \end{array}\right]$
Notes	Destabilizers X_i on top and stabilizers Z_i on bottom, by convention.	Note that application of H switches corresponding X and Z columns, which is expected since H changes the Z basis into the X basis.	Notice there is no change in the Pauli power blocks, but only the phases block because Pauli anti-commute with each other.	Same as second step.

Figure 4. The first three steps of the Deutsch-Jozsa algorithm written out in various representations, including the tableau.

balanced means that for every $b \in B$, there are precisely d^{n-1} many strings $s \in A$ such that $f(s) = b$. The task is to discern whether f is constant or balanced using queries to f on A. The algorithm described below solves a heavily restricted version of the Deutsch-Jozsa problem where $n = 1, d = 3$, and f is either constant or the identity function; a more general treatment can be found in [25, 66]. The quantum procedure below decides the problem in a single query.

Algorithm 1 (Restricted) Qudit Deutsch-Jozsa in dimension d

Require: Function f, either constant or identity
Ensure: 0 if constant, balanced otherwise
1: Initialize $\lvert 0\rangle \lvert 1\rangle$ and apply $\mathcal{F} \otimes \mathcal{F}$ to it.
2: **if** function f is constant **then**
3: $\quad U_f = X_1^j$, where $f(x) = j$, for every input x
4: **else**
5: $\quad U_f = CNOT_{0,1}$
6: **end if**
7: Apply U_f and then $(\mathcal{F}^\dagger)$.
8: Return the outcome of measuring Z_0.

Figure 3 exactly implements the circuit in 6 as an Sdim program. We have reviewed the construction of this program through syntax in the earlier sections, and now we will discuss the data this code manipulates.

4.2.1 Operations in the prime dimension qudit stabilizer tableau.

The *tableau* of a n-qudit stabilizer state $\lvert\psi\rangle$ is a table recording n of its stabilizer generators stacked in block form. As discussed in the background, the tableau and its change under unitary gates is an exact representation of the state we are simulating.

In practice, we append additional data to the tableau in the form of the "*destabilizer*" of $\lvert\psi\rangle$, or Pauli operators that

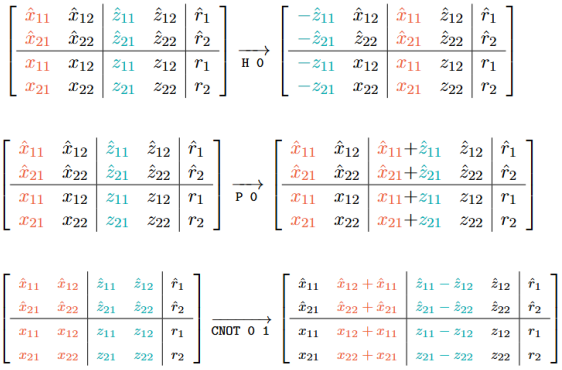


Figure 5. Tableau rewrite rules corresponding to the conjugation table in Figure 2.

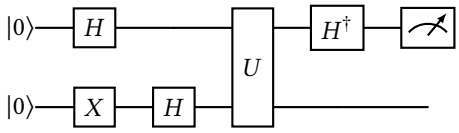


Figure 6. Quantum circuit corresponding to the code in Figure 3, which itself implements Algorithm 1.

do not stabilize the state *up to a phase*. For example, $\lvert 1\rangle$ is not stabilized by Z, but it is stabilized by $\omega^{-1}Z$, so Z is not a destabilizer generator. However, X cannot stabilize $\lvert 1\rangle$ even with additional phases, so X is a generator for other destabilizers of $\lvert 1\rangle$ (such as $X^2, X^3, \ldots$). Destabilizers were the key in seminal work [1, 32] in stabilizer simulation to achieve $O(n^2)$ time for measurement. These destabilizers form a group analogous to the state stabilizers and with an analogous n-generator representation, so in total, the tableau consists of $2n$ Pauli strings in block form. The destabilizer

block is written on top of the stabilizer block, and so altogether, the tableau takes the following form with $O(n^2)$ integers:

$$\left[\begin{array}{ccc|ccc|c} x_{0,0} & \cdots & x_{0,n-1} & z_{0,0} & \cdots & z_{0,n-1} & r_0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n-1,0} & \cdots & x_{n-1,n-1} & z_{n,0} & \cdots & z_{n-1,n-1} & r_{n-1} \\ \hline x_{n,0} & \cdots & x_{n,n} & z_{n,0} & \cdots & z_{n,n-1} & r_n \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{2n-1,1} & \cdots & x_{2n-1,n-1} & z_{2n-1,0} & \cdots & z_{2n-1,n-1} & r_{2n-1} \end{array}\right] \tag{3}$$

These tableaus are implemented with Numpy [36] arrays. We initialize the tableau as $|0\rangle^{\otimes n}$ using the destabilizer and stabilizer generators $\{X_j\}_{j=1}^n$ and $\{Z_j\}_{j=1}^n$ by convention. Evolving $|\psi\rangle$ by a gate C conjugates all stabilizer and destabilizer generators by C, and these conjugations are performing using their respective rewrite rules on the tableau. These rules detailed in 5. We only need to know how each Clifford conjugates X and Z to know how it conjugates any Pauli (and thus any Pauli tensor product).

The first three steps of the circuit in 4 are detailed in Table 4. Each row explicitly states information from the tableau or an equivalent form in another representation. Now that we can properly evolve a tableau via gates, we discuss how to extract measurements (and post-measurement states) out of it.

4.2.2 Measurement in the prime dimension qudit stabilizer tableau.

Sdim measurements are local Z measurements which suffices to perform any Pauli measurement. Given $P = U^\dagger ZU$, we may perform a P measurement by applying $U^\dagger$, performing a Z measurement, and applying U. The resulting measurement record k corresponds to collapsing to the ω^k eigenstate of P. [33]. Our general (Z) measurement algorithm is detailed in Algorithm 2.

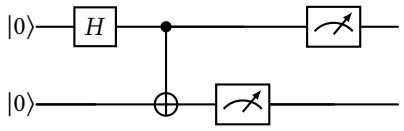


Figure 7. A simple circuit with two measurements. The first is random and followed by a deterministic measurement.

Lines 2-8 describe the procedure for a *random* measurement, while lines 10-11 detail a *deterministic* one. The procedure is a straightforward application of earlier work [1, 32]. The circuit in Figure 7 following is an example an circuit with a random measurement followed by a deterministic one.

Skipping the first two gates, we write out the tableau right before its terminal measurements:

Algorithm 2 Measurement of a qudit of (prime) dimension d

Require: Measurement index j, n-qudit state $|\psi\rangle$ in tableau as per Equation 3
Ensure: n-qudit state tableau $|\phi\rangle$, result of a Z_j measurement on $|\psi\rangle$ and measurement outcome α

1: $\alpha \leftarrow 0$
2: **if** $\exists i, j$ such that $i \geq n$ and $x_{i,j} \neq 0$ and $x_{i,l} = 0, \forall l < j$ **then**
3: **for** $k > i$ **such that** $x_{k,j} \neq 0$
4: let h be such that $h \cdot x_{i,j} + x_{k,j} \equiv 0 \pmod d$
5: row $k \leftarrow$ row $(k + h) \cdot$ row i
6: $\alpha \leftarrow$ random integer (mod d).
7: destab. row $(i - n) \leftarrow$ stab. row i.
8: stab. row $i \leftarrow$ block form of $\omega^\alpha Z_j$
9: **else**
10: **for** $i \leq n$ **such that** $x_{i,j} \neq 0$
11: $\alpha \leftarrow \alpha + r_i$
12: **end if**
13: **return** α

$$\left[\begin{array}{cc|cc|c} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \end{array}\right] \xrightarrow[\text{M 1}]{} \left[\begin{array}{cc|cc|c} 2 & 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & r \\ 0 & 0 & 2 & 1 & 0 \end{array}\right] \tag{4}$$

Measurement with random outcomes. The first Z_1 measurement is of Z_1 of the left tableau in Equation 4. The outcome is random only if the physical qudit in position j is an eigenstate (up to phase) of X or some X^aZ^b, which is only true when some $x_{i,1} \neq 0$ in the *stabilizer block* (meaning $i \geq n$). If i is the smallest such row with this property, we can assume it is the only row with the property, since we can *always* add a multiple of row block i to other non-commuting rows l until $x_{l,j} = 0$ [32] when d is prime. In this case, there is only one non-commuting row, with the offending $x_{i,1}$ value marked in red.

This row becomes the corresponding *destabilizer* in row $i - n$, since by construction, the new state is destabilized by row i. This row is finally replaced by a block form of $\omega^r Z_1$, where r is random. The total work to turn all but one row into valid stabilizers and replace the remaining row (and its corresponding destabilizer) takes $O(n^2)$ steps.

Measurement with deterministic outcomes. For the second measurement, which is of Z_0, we now simply look at the destabilizer rows of the righthand tableau. If any such row i does not commute with Z_0 (i.e. $x_{i,0} \neq 0$), take note of it. Add all phases of destabilizer rows with this property (mod d), and the result is the measurement outcome. In the case of the tableau above, only destabilizer row 0 does not commute with Z_0, so we have phase r, which is the correct

deterministic outcome. This takes $O(n)$ time to compute (in contrast to $O(n^2)$ in the qubit case [1, 15, 32]).

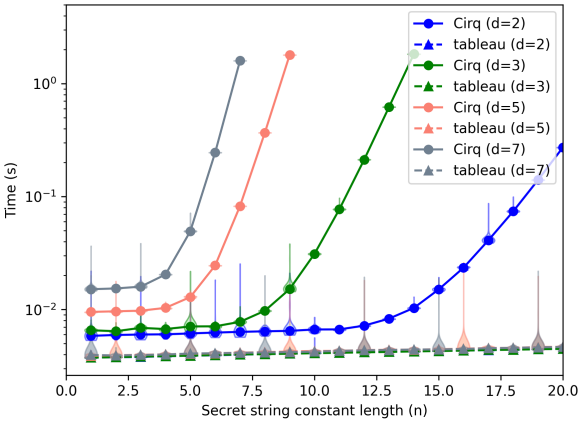


Figure 8. Cirq against Sdim on evaluating (the first shot of) Bernstein-Vazirani circuits. Each test is randomized over 100 bitstrings with the same amount of 1s, which controls the size of the most expensive subspace to measure. Note that this graph is logscale, which further demostrates that the cost of tableau simulation depends only polynomially on the system size.

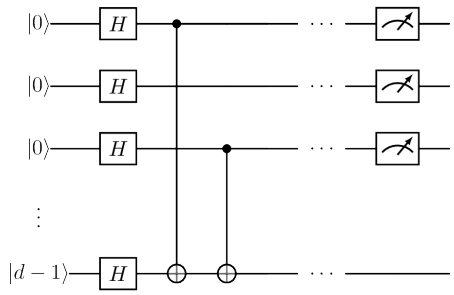


Figure 9. A qudit Bernstein-Vazirani circuit where the secret string starts with "10". For every nonzero character in the string, a qudit in that character's position entangled with the bottom ancilla via the CNOT oracle.

4.3 Validation and evaluation of single-shot performance of stabilizer tableau simulation

We cross-validate and compare Sdim's performance against Google Cirq [44], a major software framework that supports qudit circuits through state vector simulation. All tests are on the Sdim repository.

Evaluation of correctness. The correctness of our implementation is validated by the correctness of Cirq qudit circuits. We created 1000 random Clifford circuits of a fixed prime dimension at depths ranging from 5 to 1000, ran both Sdim and Cirq on these circuits over 800 shots (sampled naively via repeated shots), and recorded the measurement counts, which we will call P and Q, respectively. We verify that $\sum_{|\psi\rangle} P(|\psi\rangle) = \sum_{|\psi\rangle} Q(|\psi\rangle) = 1$. Finally, we calculate the *total variation distance* (TVD) between P and Q, where

$$\mathrm{TVD}(P,Q) = \frac{1}{2} \sum_{\text{outcome } |\psi\rangle} |P(|\psi\rangle) - Q(|\psi\rangle)| \tag{5}$$

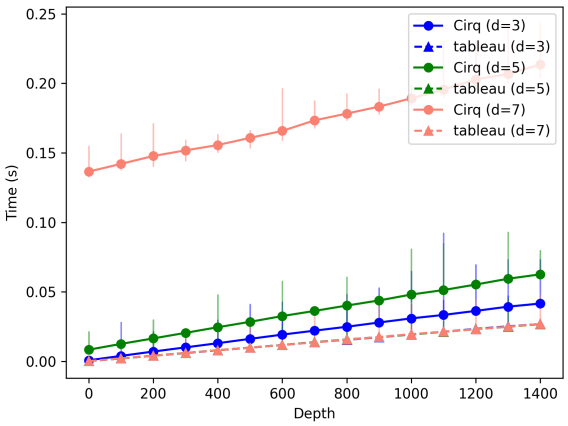
We consider the distributions P and Q matching when $\mathrm{TVD}(P,Q) < 0.2$ on all of our random circuits.

Performance evaluation via random circuits. A comparison of average runtime on random circuits is a straightforward task. Figure 1 graphs the time to calculate single shot of a quantum circuit averaged over 10 random circuits of depth 1,000 swept over dimensions $d = 3, 5, 7$. There is a clear exponential dependence on the depth of the circuit in the Cirq simulation times. This dependence is exacerbated by the qudit dimension, indicative of state vector methods involving multiplying matrices of side length d^n. On the other hand, tableau runtime is bounded by $O(n^2)$ with no apparent dependence on the dimension, reflecting the complexity of a single tableau operation depending only on its length.

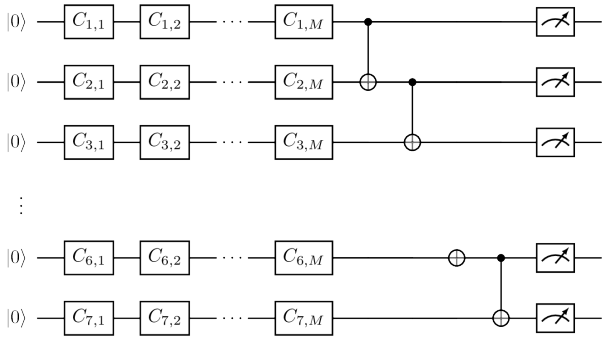
Nonetheless, the raw averages for Cirq runtimes contained outliers which likely depended on the size of the *largest set of qudits connected by CNOTs*. The complexity of a quantum measurement depends on the state space needed to describe it, and CNOTs turn single-qudit *only* measurements (that live on a d-dimensional space) into multi-qudit measurements (that live on $d^2, d^3, \ldots, d^n$ space). As a result, a proper comparison between Cirq and our stabilizer methods comes down to evaluating properly parametrized measurement performance. The *Bernstein-Vazirani* algorithm provides a circuit that allows us to precisely control the complexity of measurement.

Measurement performance evaluation via Bernstein-Vazirani. Bernstein-Vazirani (B-V) is a standard elementary quantum algorithm [6] that demonstrates the power of superposition. The task is simple: given a secret string (of d-valued dits) s and a function $f(x) = x \cdot s$ (where $\cdot$ is the dot product), determine s with minimal queries to f. The protocol to solve it is described by the circuit in Figure 9. The key takeaway is that one may control measurement complexity using the appropriate secret strings. The B-V oracle, which connects qudits to an ancilla if the qudit index correspond to a nonzero value in the bitstring (or ditstring).

In figure 8, we measure the performance of both simulators in evaluating 40-qudit B-V where the secret strings are random bitstrings with a constant number of 1s. Cirq's exponential dependence on the dimension of the largest entangled subspace completely eclipses the tableau runtime, which is only sensitive to the system size (and at most quadratic in it).



(a) Average runtime (over 100 trials at each depth M) of evaluating the circuit depicted below.



(b) Local gate test over 7 qudits. After each qudit undergoes M random Clifford gates, every qudit is entangled via a chain of $CNOT$s before a round of measurement.

Figure 10. Local gate test: runtime and circuit.

Local gate performance evaluation. Finally, we study Sdim's ability to evaluate high-depth local circuits followed entangling the all qudits and then measuring once. The results, shown in Figure 10, show the simulator runtime has a comparable linear dependence on the circuit depth, but the large leaps between the corresponding points of each line clearly demonstrate that the limiting factor here for state vector simulation is still the complexity of the terminal measurement.

5 Evaluation and application of Pauli frames for qudit FTQC benchmarking

Although stabilizer computation is faster than state vector simulation, naive repeated simulation of circuits with different noise trajectories remains prohibitively expensive. We use *Pauli frame sampling* to compute many shots of a qudit circuit, which are then benchmarked against Cirq for speed. This technique is the basis of efficient *qubit* stabilizer circuit sampling in Stim [27]. Quickly generating large ensembles of samples is a typical task in error-correcting code simulations, and so we demonstrate the Sdim's utility as such a tool by validating the logical error rates of a Cirq-simulated five-qutrit folded error detection code. Afterward, we bolster these results by simulating the error characterization procedure with parameters that were intractable in the Cirq build of the procedure.

5.1 Implementation of qudit Pauli frames for Monte Carlo Pauli error channel simulation

Pauli frames are Pauli operators aggregated together in block form. They represent random rotations and were initially proposed by Knill for qubits as a way to cope with the difficulties of real-time FTQC [42]. We use frames to simulate the randomness in non-deterministic Z-measurements without evaluating the entire circuit again. First, we take a single noiseless *reference shot* and derive the effects of noise operators using the frames. The key limitation of this method is that our noise operators are restricted to probabilistic Pauli channels, which despite their ubiquity in quantum error correction research [34, 35] struggle to model other common types of noise noise like leakage errors [51].

5.1.1 Pauli error channels. As mentioned in section 3, our simulator supports the *flip, phase,* and *depolarizing* noise channels. Flip and phase errors are pure X-type and pure Z-type errors, respectively, while depolarizing noise applies arbitrary Pauli noise. All channels act on the state by I with probability $(1 - \texttt{prob})$. The single-type error channels act non-trivially by applying X^a and Z^a, respectively, where $a \in \{1, \ldots, d-1\}$ is random. The depolarizing channel acts non-trivially by a random $X^a Z^b$, where of course $X^a Z^b \neq II$.

5.1.2 Pauli frame tableaus. The frame tableau is simply a table of stacked, *phaseless* Pauli operators.

$$\left[\begin{array}{ccc|ccc} x_{0,0} & \cdots & x_{0,n-1} & z_{0,0} & \cdots & z_{0,n-1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{s,1} & \cdots & x_{s,n} & z_{s,0} & \cdots & z_{s,n-1} \end{array}\right]$$

The length s is the number of additional samples to generate from the measurements of a single reference sample, which is evaluated *by skipping all noise channels.* Each row in this frame tableau represents the (random) data to needed to compute a distinct shot.

The procedure for initializing and evaluating a single frame (row) is as follows:

1. Initialize frame i with the block form of $Z_1^{k_1} \otimes Z_2^{k_2} \cdots \otimes Z_n^{k_n}$, where each $k_i \in \{0, \ldots, d-1\}$ is random.
2. If the frame encounters a RESET on qudit j, repeat n step 1 *only* for the j^{th} entry of each frame. Otherwise,

when evaluating Clifford gates, conjugate the frame with the rules as usual.
3. If the frame encounters a noise channel on qudit j, take the Pauli $X^k Z^l$ applied to the state and multiply it into the frame. This means $x_{i,j} \leftarrow x_{i,j} + k \mod d$ and $z_{i,j} \leftarrow z_{i,j} + l \mod d$.
4. When measuring on qudit j, record the value as $r + x_{i,j} \mod d$, where r is the value of that measurement recorded during the reference shot. Randomly initialize $z_{i,j}$ again.

These frame tableaus are Numpy arrays as well. The frame evolution of the shots above, which is just arithmetic over *columns*, is effectively vectorized by the Numpy library [36].

The Pauli frame calculates *deviations* off the reference shot whose statistics are governed by the noise gates in the circuit. However, noise gates alone do not explain how Pauli frames recover the uniform randomness seen in a Z measurement of $H|0\rangle$. This example illustrates the role of *random Z-block initializations* in steps 1, 2, and 5 above. Suppose our reference shot measures $H|0\rangle$ as k (eigenstate $|k\rangle$). We initialize our frame tableau with a random Z-block, which evolves according to the procedure above:

$$\left[\begin{array}{c|c} 0 & z_{00} \\ 0 & z_{10} \\ \vdots & \vdots \\ 0 & z_{s0} \end{array}\right] \xrightarrow[\texttt{H 0}]{} \left[\begin{array}{c|c} -z_{00} & 0 \\ -z_{10} & 0 \\ \vdots & \vdots \\ -z_{s1} & 0 \end{array}\right] \xrightarrow[\texttt{M 0}]{} \left[\begin{array}{c|c} -z_{00} & z'_{00} \\ -z_{10} & z'_{10} \\ \vdots & \vdots \\ -z_{s0} & z'_{s0} \end{array}\right]$$

where the red values are all distinct offsets to add to our reference measurement of k. It is easy to see that the distribution $\{k - z_{j0} (\mathrm{mod}\ d)\}_{0 \le j \le s}$ is the uniform random distribution as expected.

5.2 Validation and evaluation of multi-shot performance of qudit Pauli frame simulation

We examine the correctness and performance of our Pauli frame sampler.

5.2.1 Validation of Pauli frame simulation.

All tests are available on the simulator github repository as `pytest` modules.

Validation of no-error, non-deterministic measurements. Validations that Pauli frame simulation is easily achieved by sampling a random circuit (parametrized by system size n) of depth 1000 and collecting samples both by slow, manual repetition and from the Pauli frame sampler by passing in a `shots` parameter to the `simulate` method in `Program`. Similar to earlier evaluation, this will result in probability distributions P and Q over the measurement outomes, with which we validate $\mathrm{TVD}(P, Q) < 0.02$.

Validation of error channels. Noise operations are validated via the distribution of outcomes in test circuits. The procedure to test single channel error:

1. Randomly select an error probability p and dimension d, and fix a number of shots
2. Begin with an empty circuit of a single qudit. If testing Z-type errors, apply H.
3. Apply the error channel. If testing Z-type errors, apply measure in X. Otherwise, measure in Z.
4. Compute the probability distribution of outcomes P over the shots, and let Q be the distribution that assigns probability $1-p$ to 0 and $p/(d^2-1)$ to everything else. Assert that $\mathrm{TVD}(P, Q) < 0.02$.

For depolarizing noise, let Q be the distribution that assigns probability $(1 - p) + (d - 1)(p/(d^2 - 1))$ to 0 and probability $d \cdot (p/(d^2 - 1))$ for everything else. For some random Pauli O, we take O-measurements of a single qudit single depolarizing event and validate that the empirical distribution P is such that $\mathrm{TVD}(P, Q) < 0.02$.

5.2.2 Evaluation of Pauli frame performance.

In order to compare the sampling speeds between Sdim and Cirq, we run the two simulators on n-fold concatenations of the one used in Figure 10b. Each fold introduces a new round of measuring an entangled 7-qudit state, which as discussed earlier is exponentially expensive in the system size for statevector simulation. These circuits help us compare sampling complexity between simulators while also comparing the overall contribution of sampling compared to collecting a reference measurement.

We can see that in round 1 (no additional concatenation), Cirq's simulator actually outperforms the tableau at collecting shots, suggesting that the statevector sampling complexity *alone* is comparable or better than the Sdim's build of Pauli frame sampling. However, the exponential complexity of the measurements still completely overshadows any incidental sampling inefficiencies. There are small, unentangled systems where state vector computation is cheap even with *CNOT*s in the circuit, but tableau simulation is *always* cheap and certainly many orders of magnitude cheaper for large, entangled systems.

Earlier when introducing Pauli frames, we mentioned that naively repeating runs of a tableau circuit is an inefficient method of sampling. Indeed, the cost for repeatedly simulating small systems outstrips the cost of sampling from a statevector simulation.

6 Case study: Validation of a logical randomized benchmark of a five-qutrit folded code

As stated in the introduction, the simulator's sampling techniques were developed to support characterization of emerging qutrit hardware. We are motivated by the following question: can we still meaningfully protect a system from error using a code that *detects* rather than *corrects* errors? Due to the extremely limited number of qutrits and topology of the

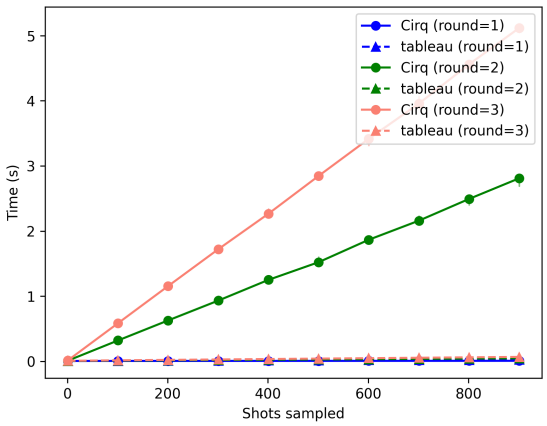


Figure 11. Rounds of concatenation of a fully entangled 7-qutrit state (identical to the ones tested in Figure 10, sampled at various shots. The run times have been averaged over 10 runs of sampling per shot count.

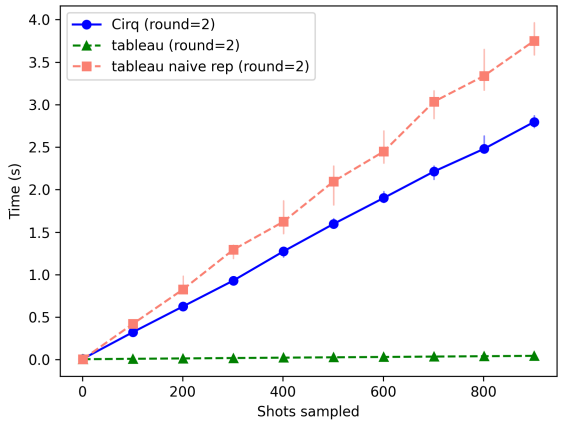


Figure 12. Time to sample the circuit from Figure 10 using Cirq, naive repeated runs with the tableau, and the Pauli frame tableau sampler.

system in question, we considered a highly non-standard *detection* code to protect 5 qutrits. The code is variant of a *folded surface code* studied in [56], but a full discussion of its construction and merits falls out of the scope of this work.

However, a concrete task within scope is to validate a Cirq-simulated error rate estimation protocol using the code parameters. Essentially, our goal is to reproduce the graph in Figure 13 in our simulator. This would validate its correctness, which is crucial because it exhibits unusual non-exponential behavior. The second goal would be to use the added performance in the tableau setting to extend the simulation past the point of state vector intractability. To that end, we discuss (Logical) Randomized Benchmarking.

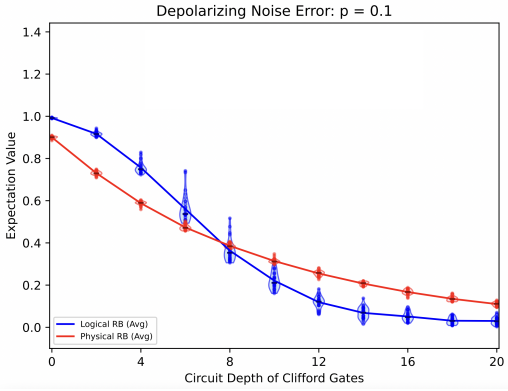


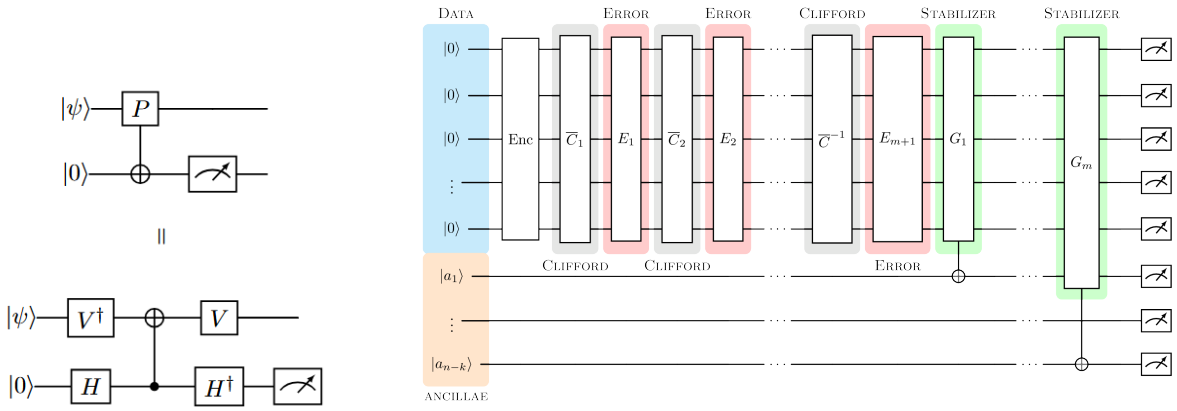
Figure 13. Cirq simulation of a *logical randomized benchmarking* protocol with *only detection* (LRB-D) against RB of an unprotected qutrit. The lines are simulated success probabilities for circuits at various depths. We expect LRB graphs of Markovian noise to be exponential, so the unusual curve is either an artifact of the modified LRB or of the code.

6.0.1 Randomized benchmarking.

The *Randomized benchmarking* (RB) [43] procedure is a Monte Carlo method that, given error rates of a noisy quantum gate set, produces an $\alpha \in [0, 1]$ such that $f(D) = B\alpha^D$ approximates the probability that a circuit of the noisy gates returns non-erroneous measurements, where B is a constant and D is circuit depth. This α may be obtained either from hardware experiments and from idealized simulations with error parameters (often both). The work in [14, 49, 50] establishes that a simulated α with access only to Clifford circuits and depolarizing noise (with an appropriately chosen failure probability) can still approximate the success probabilities of hardware-benchmarked undergoing more general errors. Finally, it is robust enough to approximate the success probability of systems with *gate-dependent* errors using an appropriately chosen gate-independent error probability [65]. Altogether, the α parameter is a powerful summary of the *average gate fidelity decay rate*, and RB is a task well suited for stabilizer simulation and Pauli frame sampling.

Each RB circuit is a random sequence of Clifford gates at a given depth, each padded by depolarizing errors, as seen in Figure 15. This sequence is followed by a (noiseless) inverse Clifford sequence, one more depolarizing event, and a terminal measurement. Assuming a fixed dimension d and gate-independent error probability p, the α is determined by the *depths* of the random circuits above and the number of random Clifford circuits at each depth. The basic procedure:

1. Generate random circuits of the type in Figure 15, parametrized by depth and the number of random circuits at each depth.
2. Simulate each circuit over a large constant number of shots. For the k^{th} random circuit of depth D, the (normalized) outcomes form a probability distribution $P_{D,k}$ over $\{0, 1, \ldots, d-1\}$.



(a) A *Non-destructive stabilizer measurement* knocks a Pauli P measurement of $|\psi\rangle$ into the ancilla without disturbing the state. Here, $V^\dagger PV = X$.

(b) Pictured above is LRB with detection only (LRB-D). As in RB, once the state is encoded into an initial (logical) state, random (logical) gates are applied and sandwiched by noise events. However, using the same gadget as on the left, we perform *stabilizer measurements* at the end via repeated conjugated CNOT gates with the ancilla as the source. These measurements tell us whether this noisy state is still in the codespace, for if not, we reject the shot entirely. Otherwise, we treat this run a regular RB run.

Figure 14. Logical randomized benchmarking with non-destructive stabilizer measurements.

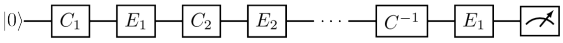


Figure 15. A typical RB circuit where C_i are the sampled Clifford gates, C^{-1} is the product of inverses of the preceding Clifford gate sequence, and E_i are the errors after each gate.

3. These probabilities yield *fidelities* $f_{D,k} = \left|\sum_{j=0}^{d} P_{D,k}(j)\omega^j\right|$.
4. Average the fidelities over k. The resulting f_D is the *average fidelity of a depth D circuit* with this noisy gate set. Work from [65] guarantees this is an exponential decay, and so a curve fitter can extract α.

6.0.2 Detection codes. Quantum (stabilizer) error correcting codes are protocols that encode k logical qudits (and their gates) across n physical ones [31]. The are defined by *codespace stabilizers*, which are Pauli operators just like state stabilizers. We can perform a *non-destructive stabilizer measurement* by following the procedure in Figure 14a, which captures the measurement outcome onto an ancilla register without disturbing the code-protected qudits. These stabilizer measurements, called *error syndromes*, are used to exactly identify errors so that they might be corrected via *recovery operations*.

In contrast, quantum *detection* codes are encodings where the stabilizer measurements are only robust enough to certify whether the state has undergone an error, meaning they cannot discern the error type. The work here considers a $[[5, 1, 2]]_3$ detection code defined by the following stabilizers:

$$\begin{aligned} G_1 &= Z_1 \otimes Z_2^{-1} \otimes Z_4^{-1} \\ G_2 &= Z_2 \otimes Z_3 \otimes Z_5^{-1} \\ G_3 &= X_1 \otimes X_2 \otimes X_3^{-1} \\ G_4 &= X_2^{-1} \otimes X_3 \otimes X_5^{-1} \end{aligned} \tag{6}$$

6.0.3 Logical randomized benchmarking. The *logical randomized benchmarking* (LRB) protocol proposed by Combes et al. [13] extends the RB framework to characterize the *logical error rate* of a code-protected system. The resulting protocol follows RB closely, except for some key modifications, detailed here:

- An n-qudit system is prepared into an initial logical state of a code. The random Clifford operators are now random *logical* Cliffords acting on the codespace, denoted with an overhead bar.
- After every noisy Clifford, the simulation is allowed to attempt to *correct* any error.
- The terminal measurement now measures a logical operator. For example, if $\overline{Z} = Z_1 Z_2$, then a $\overline{Z}$ measurement is the sum of the Z_1 and Z_2 measurement outcomes.

Oftentimes, correcting errors is a daunting practical task as the error syndromes enumerate exponentially many different recovery operations, and there are a myriad of *decoding algorithms* which quickly infer an approximate recovery operation from the syndrome data. The LRB protocol allows one to benchmark the effectiveness of a code to detect errors and a decoder to accurately solve for a recovery operation.

However, we consider a detection *only* variant, called *LRB-D*, detailed in Figure 14b. The motivation is as follows: with an early fault-tolerant quantum computing (FTQC) era

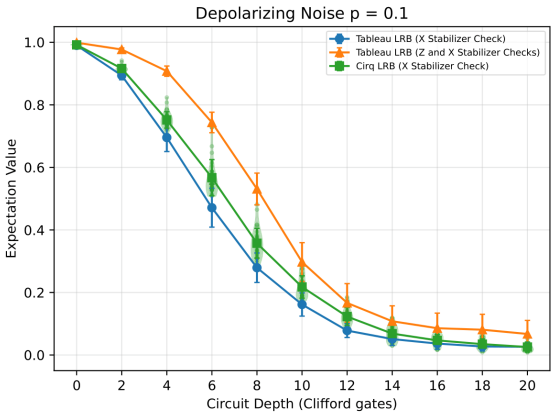


Figure 16. Sdim validation of the Cirq implementation of LRB-D (detailed in Figure 14b). We were also able to extend the original experiment and simulate measuring all codespace stabilizers with minimal impact on performance.

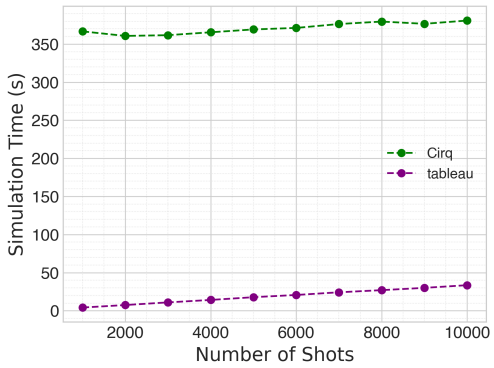


Figure 17. Simulation time per run of LRB-D (at a fixed gate-independent noise parameter, over 30 random Clifford circuits and their subcircuits) on the folded detection code versus the number of samples. Even on a 5-qutrit state, tableau methods offer significantly improved runtimes.

machine in mind, can spending a large volume of existing, error-prone resources on weaker-yet-feasible protocols (e.g detection) yield well-behaved, RB-like decay curves, and can simple postselection yield protection comparable to early fault-tolerance? The same error syndromes, which read as 0…0 when no error has been detected, allow us to *postselect* for shots that have no (detectable) error on the codespace. We then average the fidelities over these filtered shots as in LRB and extract averages.

6.0.4 Evaluation of the noise characterization. Initial efforts to study LRB-D involved a sweep of 90 error probabilities. Per probability, we sampled 30 Clifford circuits at depths $[0, 4, 8, 12, 16, 20]$ at $10,000$ shots per circuit. The high shot count is necessary for statistically significant sample sizes, as the number of shots filtered out grows exponentially with circuit depth. Consequently, measuring all the codespace stabilizers proved prohibitively in the Cirq simulation, so we restricted the postselection criteria to only syndrome data from the X-type stabilizers.

Sdim's and Cirq's curves match (within the spread of each averge), seen in Figure 16. Due to the added performance, Sdim was able to to actually measure all the codespace stabilizers as well, which led to higher fidelities, seen in orange on Figure 16. Nonetheless, the curves all drop to 0 faster than the physical error rate of a single qutrit RB.

6.0.5 Evaluation of Pauli frame monte carlo simulation performance in the case study. On Cirq, it took more than 5 days to perform the 90 error parameter sweep. On Sdim, it took less than two hours, and we were able to sample all the codespace stabilizers for postselection, leading to an increase in fidelity. The increase is shown in Figure 16 while the discrepancy in LRB runtimes between Cirq and Sdim can be seen in Figure 17.

7 Non-prime dimensions

The tableau we introduced for prime qudit stabilizer computation does not straightforwardly generalize in non-prime dimensions. For example, when d is even, Pauli $P = X^{-1}Z^{-1}$ has order $2d$, as $P^d = -I$. Furthermore, there are circuits of composite dimension for which random measurements are not uniformly random such as $CNOT_{0,1}^2$ $(F_0 \left|00\right\rangle)$ in $d = 4$. A formalism to cope with these difficulties was developed in [15]. However, the complexity of measurement is tied to being able to find the *Smith Normal Form* of an integer matrix, which can cost $O(n^4)$ time to compute.

8 Conclusion and research directions

We have built a stabilizer circuit simulator that is capable of evaluating circuits of arbitrary dimension. The correctness of the gates are exhaustively validated against statevector simulation, and the asymptotic advantage of tableau simulation against statevector simulation of a single shot (in prime dimensions) is clear even for a modest number of qudits.

Additionally, we built an efficient sampler for Pauli measurements using Pauli frames, which not only avoids repeating the more (relatively) expensive operations in the tableau but effectively vectorizes the shot computation. Despite lacking the raw efficiency of statevector sampling, our simulator's ability to evaluate large, measurement-heavy systems makes it invaluable in performing essential work in FTQC research. We used this to tackle concrete characterization tasks such as RB and LRB, which, despite the seemingly limited setting of Clifford circuits, produce fidelity calculations that can characterize hardware implementing *arbitrary* quantum gates [14, 49, 50, 65].

FTQC primitives and a maturing API. As stated in the beginning, this simulator is *only* the bare minimum. There

is a long line of work yet to be realized on the way to supporting qudit FTQC, such as developing standard workflows, pipelines, and libraries.

References

[1] Scott Aaronson and Daniel Gottesman. 2004. Improved simulation of stabilizer circuits. *Physical Review A* 70, 5 (Nov. 2004). doi:10.1103/physreva.70.052328

[2] Rajeev Acharya, Laleh Aghababaie-Beni, Igor Aleiner, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Nikita Astrakhantsev, Juan Atalaya, Ryan Babbush, Dave Bacon, Brian Ballard, Joseph C. Bardin, Johannes Bausch, Andreas Bengtsson, Alexander Bilmes, Sam Blackwell, Sergio Boixo, Gina Bortoli, Alexandre Bourassa, Jenna Bovaird, Leon Brill, Michael Broughton, David A. Browne, Brett Buchea, Bob B. Buckley, David A. Buell, Tim Burger, Brian Burkett, Nicholas Bushnell, Anthony Cabrera, Juan Campero, Hung-Shen Chang, Yu Chen, Zijun Chen, Ben Chiaro, Desmond Chik, Charina Chou, Jahan Claes, Agnetta Y. Cleland, Josh Cogan, Roberto Collins, Paul Conner, William Courtney, Alexander L. Crook, Ben Curtin, Sayan Das, Alex Davies, Laura De Lorenzo, Dripto M. Debroy, Sean Demura, Michel Devoret, Agustin Di Paolo, Paul Donohoe, Ilya Drozdov, Andrew Dunsworth, Clint Earle, Thomas Edlich, Alec Eickbusch, Aviv Moshe Elbag, Mahmoud Elzouka, Catherine Erickson, Lara Faoro, Edward Farhi, Vinicius S. Ferreira, Leslie Flores Burgos, Ebrahim Forati, Austin G. Fowler, Brooks Foxen, Suhas Ganjam, Gonzalo Garcia, Robert Gasca, ÃĽlie Genois, William Giang, Craig Gidney, Dar Gilboa, Raja Gosula, Alejandro Grajales Dau, Dietrich Graumann, Alex Greene, Jonathan A. Gross, Steve Habegger, John Hall, Michael C. Hamilton, Monica Hansen, Matthew P. Harrigan, Sean D. Harrington, Francisco J. H. Heras, Stephen Heslin, Paula Heu, Oscar Higgott, Gordon Hill, Jeremy Hilton, George Holland, Sabrina Hong, Hsin-Yuan Huang, Ashley Huff, William J. Huggins, Lev B. Ioffe, Sergei V. Isakov, Justin Iveland, Evan Jeffrey, Zhang Jiang, Cody Jones, Stephen Jordan, Chaitali Joshi, Pavol Juhas, Dvir Kafri, Hui Kang, Amir H. Karamlou, Kostyantyn Kechedzhi, Julian Kelly, Trupti Khaire, Tanuj Khattar, Mostafa Khezri, Seon Kim, Paul V. Klimov, Andrey R. Klots, Bryce Kobrin, Pushmeet Kohli, Alexander N. Korotkov, Fedor Kostritsa, Robin Kothari, Borislav Kozlovskii, John Mark Kreikebaum, Vladislav D. Kurilovich, Nathan Lacroix, David Landhuis, Tiano Lange-Dei, Brandon W. Langley, Pavel Laptev, Kim-Ming Lau, LoÃŕck Le Guevel, Justin Ledford, Kenny Lee, Yuri D. Lensky, Shannon Leon, Brian J. Lester, Wing Yan Li, Yin Li, Alexander T. Lill, Wayne Liu, William P. Livingston, Aditya Locharla, Erik Lucero, Daniel Lundahl, Aaron Lunt, Sid Madhuk, Fionn D. Malone, Ashley Maloney, Salvatore MandrÃą, Leigh S. Martin, Steven Martin, Orion Martin, Cameron Maxfield, Jarrod R. McClean, Matt McEwen, Seneca Meeks, Anthony Megrant, Xiao Mi, Kevin C. Miao, Amanda Mieszala, Reza Molavi, Sebastian Molina, Shirin Montazeri, Alexis Morvan, Ramis Movassagh, Wojciech Mruczkiewicz, Ofer Naaman, Matthew Neeley, Charles Neill, Ani Nersisyan, Hartmut Neven, Michael Newman, Jiun How Ng, Anthony Nguyen, Murray Nguyen, Chia-Hung Ni, Thomas E. O'Brien, William D. Oliver, Alex Opremcak, Kristoffer Ottosson, Andre Petukhov, Alex Pizzuto, John Platt, Rebecca Potter, Orion Pritchard, Leonid P. Pryadko, Chris Quintana, Ganesh Ramachandran, Matthew J. Reagor, David M. Rhodes, Gabrielle Roberts, Eliott Rosenberg, Emma Rosenfeld, Pedram Roushan, Nicholas C. Rubin, Negar Saei, Daniel Sank, Kannan Sankaragomathi, Kevin J. Satzinger, Henry F. Schurkus, Christopher Schuster, Andrew W. Senior, Michael J. Shearn, Aaron Shorter, Noah Shutty, Vladimir Shvarts, Shraddha Singh, Volodymyr Sivak, Jindra Skruzny, Spencer Small, Vadim Smelyanskiy, W. Clarke Smith, Rolando D. Somma, Sofia Springer, George Sterling, Doug Strain, Jordan Suchard, Aaron Szasz, Alex Sztein, Douglas Thor, Alfredo Torres, M. Mert Torunbalci, Abeer Vaishnav, Justin Vargas, Sergey Vdovichev, Guifre Vidal, Benjamin Villalonga, Catherine Vollgraff Heidweiller, Steven Waltman, Shannon X. Wang, Brayden Ware, Kate Weber, Theodore White, Kristi Wong, Bryan W. K. Woo, Cheng Xing, Z. Jamie Yao, Ping Yeh, Bicheng Ying, Juhwan Yoo, Noureldin Yosri, Grayson Young, Adam Zalcman, Yaxing Zhang, Ningfeng Zhu, and Nicholas Zobrist. 2024. Quantum error correction below the surface code threshold. arXiv:2408.13687 [quant-ph] https://arxiv.org/abs/2408.13687

[3] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510. doi:10.1038/s41586-019-1666-5

[4] Jonathan M. Baker, Casey Duckering, and Frederic T. Chong. 2020. Efficient Quantum Circuit Decompositions via Intermediate Qudits. In *2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL)*. 303–308. doi:10.1109/ISMVL49045.2020.9345604

[5] Jonathan M. Baker, Casey Duckering, Pranav Gokhale, Natalie C. Brown, Kenneth R. Brown, and Frederic T. Chong. 2020. Improved Quantum Circuits via Intermediate Qutrits. *ACM Transactions on Quantum Computing* 1, 1, Article 2 (oct 2020), 25 pages. doi:10.1145/3406309

[6] Ethan Bernstein and Umesh Vazirani. 1997. Quantum Complexity Theory. *SIAM J. Comput.* 26, 5 (1997), 1411–1473. arXiv:https://doi.org/10.1137/S0097539796300921 doi:10.1137/S0097539796300921

[7] M. S. Blok, V. V. Ramasesh, T. Schuster, K. OâĂŹBrien, J. M. Kreikebaum, D. Dahlen, A. Morvan, B. Yoshida, N. Y. Yao, and I. Siddiqi. 2021. Quantum Information Scrambling on a Superconducting Qutrit Processor. *Physical Review X* 11, 2 (April 2021). doi:10.1103/physrevx.11.021010

[8] Earl T. Campbell. 2014. Enhanced Fault-Tolerant Quantum Computing in *d*-Level Systems. *Phys. Rev. Lett.* 113 (Dec 2014), 230501. Issue 23. doi:10.1103/PhysRevLett.113.230501

[9] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. 2017. Roads towards fault-tolerant universal quantum computation. *Nature* 549, 7671 (2017), 172–179. doi:10.1038/nature23460

[10] Shuxiang Cao, Mustafa Bakr, Giulio Campanaro, Simone D Fasciati, James Wills, Deep Lall, Boris Shteynas, Vivek Chidambaram, Ivan Rungger, and Peter Leek. 2024. Emulating two qubits with a four-level transmon qudit for variational quantum algorithms. *Quantum Science and Technology* 9, 3 (apr 2024), 035003. doi:10.1088/2058-9565/ad37d4

[11] Turbasu Chatterjee, Arnav Das, Subhayu Kumar Bala, Amit Saha, Anupam Chattopadhyay, and Amlan Chakrabarti. 2023. QuDiet: A classical simulation platform for qubitâĂŘqudit hybrid quantum systems. *IET Quantum Communication* 4, 4 (March 2023), 167âĂŞ180. doi:10.1049/qtc2.12058

[12] Yulin Chi, Jieshan Huang, Zhanchuan Zhang, Jun Mao, Zinan Zhou, Xiaojiong Chen, Chonghao Zhai, Jueming Bao, Tianxiang Dai, Huihong Yuan, Ming Zhang, Daoxin Dai, Bo Tang, Yan Yang, Zhihua Li, Yunhong Ding, Leif K. Oxenløwe, Mark G. Thompson, Jeremy L. O'Brien, Yan Li, Qihuang Gong, and Jianwei Wang. 2022. A programmable qudit-based quantum processor. *Nature Communications* 13, 1 (2022), 1166. doi:10.1038/s41467-022-28767-x

[13] Joshua Combes, Christopher Granade, Christopher Ferrie, and Steven T. Flammia. 2017. Logical Randomized Benchmarking. arXiv:1702.03688 [quant-ph] https://arxiv.org/abs/1702.03688

[14] Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. 2009. Exact and approximate unitary 2-designs and their application to fidelity estimation. *Physical Review A* 80, 1 (July 2009). doi:10.1103/physreva.80.012304

[15] Niel de Beaudrap. 2013. A linearized stabilizer formalism for systems of finite dimension. *Quantum Information and Computation* 13, 1,2 (Jan. 2013), 73âĂŞ115. doi:10.26421/qic13.1-2-6

[16] Tiago de Souza Farias, Lucas Friedrich, and Jonas Maziero. 2024. QuForge: A Library for Qudits Simulation. arXiv:2409.17716 [quant-ph] https://arxiv.org/abs/2409.17716

[17] Ronald de Wolf. 2023. Quantum Computing: Lecture Notes. arXiv:1907.09415 [quant-ph] https://arxiv.org/abs/1907.09415

[18] Yannick Deller, Sebastian Schmitt, Maciej Lewenstein, Steve Lenk, Marika Federer, Fred Jendrzejewski, Philipp Hauke, and Valentin Kasper. 2023. Quantum approximate optimization algorithm for qudit systems. *Phys. Rev. A* 107 (Jun 2023), 062410. Issue 6. doi:10.1103/PhysRevA.107.062410

[19] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. 2002. Topological quantum memory. *J. Math. Phys.* 43, 9 (09 2002), 4452–4505. arXiv:https://pubs.aip.org/aip/jmp/article-pdf/43/9/4452/19183135/4452_1_online.pdf doi:10.1063/1.1499754

[20] Advait Deshpande. 2022. Assessing the quantum-computing landscape. *Commun. ACM* 65, 10 (Sept. 2022), 57âĂŞ65. doi:10.1145/3524109

[21] Simon J Devitt, William J Munro, and Kae Nemoto. 2013. Quantum error correction for beginners. *Reports on Progress in Physics* 76, 7 (jun 2013), 076001. doi:10.1088/0034-4885/76/7/076001

[22] Yongshan Ding, Adam Holmes, Ali Javadi-Abhari, Diana Franklin, Margaret Martonosi, and Frederic T. Chong. 2018. Magic-state functional units: mapping and scheduling multi-level distillation circuits for fault-tolerant quantum architectures. In *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture* (Fukuoka, Japan) *(MICRO-51)*. IEEE Press, 828âĂŞ840. doi:10.1109/MICRO.2018.00072

[23] Thomas Durt, Nicolas J. Cerf, Nicolas Gisin, and Marek Żukowski. 2003. Security of quantum key distribution with entangled qutrits. *Phys. Rev. A* 67 (Jan 2003), 012311. Issue 1. doi:10.1103/PhysRevA.67.012311

[24] Chris Edwards. 2022. Error Control Begins to Shape Quantum Architectures. *Commun. ACM* 66, 1 (Dec. 2022), 13âĂŞ15. doi:10.1145/3570518

[25] Yale Fan. 2007. A Generalization of the Deutsch-Jozsa Algorithm to Multi-Valued Quantum Logic . In *37th International Symposium on Multiple-Valued Logic (ISMVL'07)*. IEEE Computer Society, Los Alamitos, CA, USA, 12. doi:10.1109/ISMVL.2007.3

[26] A. Fedorov, L. Steffen, M. Baur, M. P. da Silva, and A. Wallraff. 2012. Implementation of a Toffoli gate with superconducting circuits. *Nature* 481, 7380 (2012), 170–172. doi:10.1038/nature10713

[27] Craig Gidney. 2021. Stim: a fast stabilizer circuit simulator. *Quantum* 5 (July 2021), 497. doi:10.22331/q-2021-07-06-497

[28] Pranav Gokhale, Jonathan M. Baker, Casey Duckering, Natalie C. Brown, Kenneth R. Brown, and Frederic T. Chong. 2019. Asymptotic improvements to quantum circuits via qutrits. In *Proceedings of the 46th International Symposium on Computer Architecture* (Phoenix, Arizona) *(ISCA '19)*. Association for Computing Machinery, New York, NY, USA, 554âĂŞ566. doi:10.1145/3307650.3322253

[29] Noah Goss, Alexis Morvan, Brian Marinelli, Bradley K. Mitchell, Long B. Nguyen, Ravi K. Naik, Larry Chen, Christian Jünger, John Mark Kreikebaum, David I. Santiago, Joel J. Wallman, and Irfan Siddiqi. 2022. High-fidelity qutrit entangling gates for superconducting circuits. *Nature Communications* 13, 1 (2022), 7481. doi:10.1038/s41467-022-34851-z

[30] Daniel Gottesman. 1998. The Heisenberg Representation of Quantum Computers. arXiv:quant-ph/9807006 [quant-ph] https://arxiv.org/abs/quant-ph/9807006

[31] Daniel Gottesman. 1998. Theory of fault-tolerant quantum computation. *Physical Review A* 57, 1 (Jan. 1998), 127âĂŞ137. doi:10.1103/physreva.57.127

[32] Daniel Gottesman. 1999. Fault-Tolerant Quantum Computation with Higher-Dimensional Systems. In *Quantum Computing and Quantum Communications*, Colin P. Williams (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 302–313.

[33] Daniel Gottesman. 2024. Surviving as a Quantum Computer in a Classical World. URL: https://www.cs.umd.edu/class/spring2024/cmsc858G/. Last visited on 2024/10/28.

[34] Mauricio Gutiérrez and Kenneth R. Brown. 2015. Comparison of a quantum error-correction threshold for exact and approximate errors. *Phys. Rev. A* 91 (Feb 2015), 022335. Issue 2. doi:10.1103/PhysRevA.91.022335

[35] Mauricio Gutiérrez, Lukas Svec, Alexander Vargo, and Kenneth R. Brown. 2013. Approximation of realistic errors by Clifford channels and Pauli measurements. *Phys. Rev. A* 87 (Mar 2013), 030302. Issue 3. doi:10.1103/PhysRevA.87.030302

[36] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. doi:10.1038/s41586-020-2649-2

[37] Torsten Hoefler, Thomas Häner, and Matthias Troyer. 2023. Disentangling Hype from Practicality: On Realistically Achieving Quantum Advantage. *Commun. ACM* 66, 5 (April 2023), 82âĂŞ87. doi:10.1145/3571725

[38] Pavel Hrmo, Benjamin Wilhelm, Lukas Gerster, Martin W. van Mourik, Marcus Huber, Rainer Blatt, Philipp Schindler, Thomas Monz, and Martin Ringbauer. 2023. Native qudit entanglement in a trapped ion quantum processor. *Nature Communications* 14, 1 (2023), 2242. doi:10.1038/s41467-023-37375-2

[39] Fei Hua, Yanhao Chen, Yuwei Jin, Chi Zhang, Ari Hayes, Youtao Zhang, and Eddy Z. Zhang. 2021. AutoBraid: A Framework for Enabling Efficient Surface Code Communication in Quantum Computing. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture* (Virtual Event, Greece) *(MICRO '21)*. Association for Computing Machinery, New York, NY, USA, 925âĂŞ936. doi:10.1145/3466752.3480072

[40] Ali Javadi-Abhari, Pranav Gokhale, Adam Holmes, Diana Franklin, Kenneth R. Brown, Margaret Martonosi, and Frederic T. Chong. 2017. Optimized surface code communication in superconducting quantum computers. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture* (Cambridge, Massachusetts) *(MICRO-50 '17)*. Association for Computing Machinery, New York, NY, USA, 692âĂŞ705. doi:10.1145/3123939.3123949

[41] James Keppens, Quinten Eggerickx, Vukan Levajac, George Simion, and Bart SorÃľe. 2025. Qudit vs. Qubit: Simulated performance of error correction codes in higher dimensions. arXiv:2502.05992 [quant-ph] https://arxiv.org/abs/2502.05992

[42] E. Knill. 2005. Quantum computing with realistically noisy devices. *Nature* 434, 7029 (March 2005), 39âĂŞ44. doi:10.1038/nature03350

[43] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. 2008. Randomized benchmarking of quantum gates. *Physical Review A* 77, 1 (Jan. 2008). doi:10.1103/physreva.77.012307

[44] Svyatoslav Kushnarev and Hassan Jameel Asghar. 2025. The Qudit Cirq Library: An Extension of Google's Cirq Library for Qudits. arXiv:2501.07812 [quant-ph] https://arxiv.org/abs/2501.07812

[45] B. P. Lanyon, T. J. Weinhold, N. K. Langford, J. L. O'Brien, K. J. Resch, A. Gilchrist, and A. G. White. 2008. Manipulating Biphotonic Qutrits. *Phys. Rev. Lett.* 100 (Feb 2008), 060504. Issue 6. doi:10.1103/PhysRevLett.100.060504

[46] Andrew Litteken, Jonathan M. Baker, and Frederic T. Chong. 2022. Communication Trade Offs in Intermediate Qudit Circuits. In *2022 IEEE 52nd International Symposium on Multiple-Valued Logic (ISMVL)*. 43–49. doi:10.1109/ISMVL52857.2022.00014

[47] Andrew Litteken, Lennart Maximilian Seifert, Jason D. Chadwick, Natalia Nottingham, Tanay Roy, Ziqian Li, David Schuster, Frederic T. Chong, and Jonathan M. Baker. 2023. Dancing the Quantum Waltz: Compiling Three-Qubit Gates on Four Level Architectures. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA âĂŹ23)*. ACM. doi:10.1145/3579371.3589106

[48] Kai Luo, Wenhui Huang, Ziyu Tao, Libo Zhang, Yuxuan Zhou, Ji Chu, Wuxin Liu, Biying Wang, Jiangyu Cui, Song Liu, Fei Yan, Man-Hong Yung, Yuanzhen Chen, Tongxing Yan, and Dapeng Yu. 2023. Experimental Realization of Two Qutrits Gate with Tunable Coupling in Superconducting Circuits. *Phys. Rev. Lett.* 130 (Jan 2023), 030603. Issue 3. doi:10.1103/PhysRevLett.130.030603

[49] Easwar Magesan, J. M. Gambetta, and Joseph Emerson. 2011. Scalable and Robust Randomized Benchmarking of Quantum Processes. *Physical Review Letters* 106, 18 (May 2011). doi:10.1103/physrevlett.106.180504

[50] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. 2012. Characterizing quantum gates via randomized benchmarking. *Physical Review A* 85, 4 (April 2012). doi:10.1103/physreva.85.042311

[51] Hidetaka Manabe, Yasunari Suzuki, and Andrew S. Darmawan. 2025. Efficient Simulation of Leakage Errors in Quantum Error Correcting Codes Using Tensor Network Methods. arXiv:2308.08186 [quant-ph] https://arxiv.org/abs/2308.08186

[52] Kevin Mato, Martin Ringbauer, Lukas Burgholzer, and Robert Wille. 2024. MQT Qudits: A Software Framework for Mixed-Dimensional Quantum Computing. arXiv:2410.02854 [quant-ph] https://arxiv.org/abs/2410.02854

[53] Kevin Mato, Martin Ringbauer, Stefan Hillmich, and Robert Wille. 2022. Adaptive Compilation of Multi-Level Quantum Operations . In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE Computer Society, Los Alamitos, CA, USA, 484–491. doi:10.1109/QCE53715.2022.00070

[54] N.D. Mermin. 2007. *Quantum Computer Science: An Introduction*. Cambridge University Press. https://books.google.com/books?id=q2S9APxFdUQC

[55] Don Monroe. 2024. More Efficient Fault-Tolerant Quantum Computing. *Commun. ACM* 67, 5 (May 2024), 26âĂŞ28. doi:10.1145/3640350

[56] Jonathan E. Moussa. 2016. Transversal Clifford gates on folded surface codes. *Physical Review A* 94, 4 (Oct. 2016). doi:10.1103/physreva.94.042316

[57] Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.

[58] National Academies of Sciences Engineering and Medicine. 2019. *Quantum Computing: Progress and Prospects*. The National Academies Press, Washington, DC. doi:10.17226/25196

[59] John Preskill. 1998. Lecture notes for physics 229: Quantum information and computation. (1998).

[60] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (Aug. 2018), 79. doi:10.22331/q-2018-08-06-79

[61] T. C. Ralph, K. J. Resch, and A. Gilchrist. 2007. Efficient Toffoli gates using qudits. *Phys. Rev. A* 75 (Feb 2007), 022313. Issue 2. doi:10.1103/PhysRevA.75.022313

[62] Martin Ringbauer, Michael Meth, Lukas Postler, Roman Stricker, Rainer Blatt, Philipp Schindler, and Thomas Monz. 2022. A universal qudit quantum processor with trapped ions. *Nature Physics* 18, 9 (2022), 1053–1057. doi:10.1038/s41567-022-01658-0

[63] Lennart Maximilian Seifert, Jason Chadwick, Andrew Litteken, Frederic T. Chong, and Jonathan M. Baker. 2022. Time-Efficient Qudit Gates through Incremental Pulse Re-seeding . In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE Computer Society, Los Alamitos, CA, USA, 304–313. doi:10.1109/QCE53715.2022.00051

[64] S. P. Walborn, D. S. Lemelle, M. P. Almeida, and P. H. Souto Ribeiro. 2006. Quantum Key Distribution with Higher-Order Alphabets Using Spatially Encoded Qudits. *Phys. Rev. Lett.* 96 (Mar 2006), 090501. Issue 9. doi:10.1103/PhysRevLett.96.090501

[65] Joel J. Wallman. 2018. Randomized benchmarking with gate-dependent noise. *Quantum* 2 (Jan. 2018), 47. doi:10.22331/q-2018-01-29-47

[66] Yuchen Wang, Zixuan Hu, Barry C. Sanders, and Sabre Kais. 2020. Qudits and High-Dimensional Quantum Computing. *Frontiers in Physics* 8 (Nov. 2020). doi:10.3389/fphy.2020.589504

[67] Jordi R. Weggemans, Alexander Urech, Alexander Rausch, Robert Spreeuw, Richard Boucherie, Florian Schreck, Kareljan Schoutens, Jiří Minář, and Florian Speelman. 2022. Solving correlation clustering with QAOA and a Rydberg qudit system: a full-stack approach. *Quantum* 6 (April 2022), 687. doi:10.22331/q-2022-04-13-687

[68] Anbang Wu, Gushu Li, Hezi Zhang, Gian Giacomo Guerreschi, Yufei Ding, and Yuan Xie. 2022. A synthesis framework for stitching surface code with superconducting quantum devices. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 337âĂŞ350. doi:10.1145/3470496.3527381

[69] Qian Xu, J. Pablo Bonilla Ataides, Christopher A. Pattison, Nithin Raveendran, Dolev Bluvstein, Jonathan Wurtz, Bane Vasić, Mikhail D. Lukin, Liang Jiang, and Hengyun Zhou. 2024. Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays. *Nature Physics* 20, 7 (2024), 1084–1090. doi:10.1038/s41567-024-02479-z

[70] M. A. Yurtalan, J. Shi, M. Kononenko, A. Lupascu, and S. Ashhab. 2020. Implementation of a Walsh-Hadamard Gate in a Superconducting Qutrit. *Phys. Rev. Lett.* 125 (Oct 2020), 180504. Issue 18. doi:10.1103/PhysRevLett.125.180504

Appendix A Step-by-step illustration of the qudit Deutsch-Jozsa algorithm

Source code	`c = Circuit(2, 3)`	`c.add_gate("H", 0)`	`c.add_gate("X", 1)`	`c.add_gate("H", 1)`
Gate operation	Initialization	$\mathcal{F} \otimes I$	$I \otimes X$	$I \otimes \mathcal{F}$
Gate unitary	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes I$	$\frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix} \otimes I$	$I \otimes \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$I \otimes \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix}$
Statevector	$\lvert 0\rangle \otimes \lvert 0\rangle$	$\lvert +\rangle \otimes \lvert 0\rangle$	$\lvert +\rangle \otimes \lvert 1\rangle$	$\lvert +\rangle \otimes \lvert \omega\rangle$
Stabilizers	$\begin{matrix} II & IZ & IZ^2 \\ ZI & ZZ & ZZ^2 \\ Z^2I & Z^2Z & Z^2Z^2 \end{matrix}$	$\begin{matrix} II & IZ & IZ^2 \\ X^2I & X^2Z & X^2Z^2 \\ XI & XZ & XZ^2 \end{matrix}$	$\begin{matrix} II & I\omega^2Z & I\omega Z^2 \\ X^2I & X^2\omega^2Z & X^2\omega Z^2 \\ XI & X\omega^2Z & X\omega Z^2 \end{matrix}$	$\begin{matrix} II & I\omega^2X^2 & I\omega X \\ X^2I & X^2\omega^2X^2 & X^2\omega X \\ XI & X\omega^2X^2 & X\omega X \end{matrix}$
Destabilizer generators	$\{XI, IX\}$	$\{ZI, IX\}$	$\{ZI, IX\}$	$\{ZI, IZ\}$
Stabilizer generators	$\{ZI, IZ\}$	$\{X^2I, IZ\}$	$\{X^2I, \omega^2 IZ\}$	$\{X^2I, \omega^2 IX^2\}$
Tableau	$\left[\begin{array}{cc\|cc\|c} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$	$\left[\begin{array}{cc\|cc\|c} \underline{0} & 0 & \underline{1} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline \underline{2} & 0 & \underline{0} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$	$\left[\begin{array}{cc\|cc\|c} 0 & 0 & 1 & 0 & 0 \\ 0 & \underline{1} & 0 & \underline{0} & 0 \\ \hline 2 & 0 & 0 & 0 & 0 \\ 0 & \underline{0} & 0 & \underline{1} & \underline{2} \end{array}\right]$	$\left[\begin{array}{cc\|cc\|c} 0 & 0 & 1 & 0 & 0 \\ 0 & \underline{0} & 0 & \underline{1} & 0 \\ \hline 2 & 0 & 0 & 0 & 0 \\ 0 & \underline{2} & 0 & \underline{0} & 2 \end{array}\right]$
Notes	Destabilizers X_i on top and stabilizers Z_i on bottom, by convention.	Note that application of H switches corresponding X and Z columns, which is expected since H changes the Z basis into the X basis.	Notice there is no change in the Pauli power blocks, but only the phases block because Pauli anti-commute with each other.	Same as second step.

Appendix B Prime dimension measurement reduces to Gaussian elimination

Consider tableau A for state $|\psi\rangle$ *without* its destabilizer generators, where we regard it as a matrix over $\mathbb{F}_d$:

$$A = \left[\begin{array}{ccc|ccc|c} x_{1,1} & \dots & x_{1,n} & z_{1,1} & \dots & z_{1,n} & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n,1} & \dots & x_{n,n} & z_{n,1} & \dots & z_{n,n} & r_n \end{array}\right] = \begin{bmatrix} - - g_1 - - \\ \vdots \\ - - g_n - - \end{bmatrix}$$

In order to perform a Z_j measurement, we first ask whether Z_j is in the stabilizer group $S(|\psi\rangle)$. If so, then the following equation has a solution via an assignment to variables $y_1, \dots, y_n$:

$$Z_j = \omega^l \prod_{j=1}^{n} g_j^{y_j}$$

where $l \in \{0, \dots, d-1\}$. If there is a solution, then by definition $\omega^{-l} Z_j \in S(|\psi\rangle)$. On the other hand, if no solution exists, then $|\psi\rangle$ is not an eigenstate of Z_j, and the measurement outcome is random. We can formulate this as a linear algebra problem. Let:

$$\vec{y} = \left[y_1, y_2, \dots, y_n\right]^\mathsf{T} \in \mathbb{F}_d^n$$

and write our measurement operator Z_j in the generator encoding:

$$\vec{z_j} = \left[0, \dots, 0 \mid 0, \dots 1, \dots, 0 \mid 0\right]^\mathsf{T} = e_{n+j}^\mathsf{T} \in \mathbb{F}_d^{2n+1}$$

Now, the following:

$$A^\mathsf{T} \vec{y} = \vec{z_j}$$

is an encoding of our first problem. We can easily solve this by performing Gaussian elimination on the augmented matrix:

$$\left[\begin{array}{c|c} A^\mathsf{T} & z_j \end{array}\right]$$

Let $B = \text{rref}(A)$. After Gaussian elimination, our augmented system looks like:

$$\left[\begin{array}{c|c} B^\mathsf{T} & z' \end{array}\right]$$

If this form gives us a solution to the y_j variables, then our outcome is deterministically $-l$, as $|\psi\rangle$ is an ω^{-l}-eigenstate of Z_j (or equivalently, $\omega^{-l} Z_j$ stabilizes $|\psi\rangle$). We can naively solve for $-l$ by directly computing $\prod_{j=1}^n g_j^{y_j} = \omega^{-l} Z_j$.

If the form provides no solution, B still gives us the relevant information to perform the random measurement. For any $a, b \in \{1, \dots, n\}$, notice that $g_a g_b \in S(|\psi\rangle)$ and $g_a^k \in S(|\psi\rangle)$, where $k \in \{0, \dots, d-1\}$. These operations correspond exactly to the steps in Gaussian elimination, so this tells us that

$$B = \begin{bmatrix} - - g_1' - - \\ \vdots \\ - - g_n' - - \end{bmatrix}$$

is still a tableau for $|\psi\rangle$. However, each g_j' acts on mutually exclusive qudits. For example, if $g_j' = Z_1 Z_2$, then *no other* generator in B is made of (non-identity) operators that act on qudits 1 and 2. Since this measurement is random, there must be a unique generator g_q' that includes a power of X_j. Replacing g_q' with $\omega^r Z_j$, where $r \in \{0, \dots, d-1\}$ is random, gives us a tableau for our post-measurement state. Of course, our measurement outcome was r.

For $d = 2$, evaluating $\prod_{j=1}^n g_j^{y_j}$ may involve multiplying powers of the imaginary unit i, and the rowsum protocol in [1] exactly tracks those contributions to the phase calculation. However, this technical detail does not appear for odd prime d [32].

Appendix C Non-prime stabilizer tableau simulation

The tableau we introduced for prime qudit stabilizer computation does not cleanly generalize for non-prime dimensions. For example, when d is even, Pauli $P = X^{-1}Z^{-1}$ has order $2d$, as $P^d = -I$. These differences manifest both during and after measurement; while $d = 2$ is prime, we observe that as an even dimension, deterministic measurements are more difficult due to factors of i coming out in the phase.

A formalism to cope with these difficulties was developed in [15]. Here, we will outline the essential differences from the prime case. All difficulties arise from the even cases, so we will assume that $d > 2$ is even.

C.1 Representation of non-prime stabilizer generators

The essential difference between the prime and non-prime cases is the *power of d* that show up in the phases of the stabilizer. This is seen most simply again when $d = 2$, where i, a 4th of unity, appears in the phases. To deal with this, we represent a Pauli stabilizer $Z^a X^b$ as a *Weyl operator* $W_{a,b} = \tau^{-ab} Z^a X^b$, where $\tau = \exp\left((i\pi(d^2+1))/d\right)$. We can extend this to a product $Z^{a_1}X^{b_1} \otimes \cdots \otimes Z^{a_n}X^{b_n}$ by writing:

$$W_{\vec{a},\vec{b}} = \tau^{-\vec{a}\cdot\vec{b}} (Z^{a_1} \otimes \cdots \otimes Z^{a_n})(X^{b_1} \otimes \cdots \otimes X^{b_n})$$

where $\vec{a} = [a_1, \dots, a_n]$, $\vec{b} = [b_1, \dots, b_n]$, and $\vec{a} \cdot \vec{b}$ is the dot product. While this helps us deal with the factors of d^2, we still have problems where $W_{0,1} = -W_{d,1}$ when d is even, but this cannot be true since $W_{d,1}$ is non-zero. The second modification is to lift all our our modular arithmetic to d', which is simply d when d is odd and $2d$ when d is even. With these modifications in mind, our Weyl operators all have order d'.

C.2 Operations in the non-prime dimension qudit stabilizer tableau

We do not track any generalization of destabilizer generators, which just leaves with X, Z, and phase blocks, which encode the integers defining $W_{\vec{a},\vec{b}}$. Here, we have $2n$ such operators $W_{\vec{a_1},\vec{b_1}}, \dots, W_{\vec{a_{2n}},\vec{b_{2n}}}$. Note that each $\vec{a_k}$ is a vector of numbers,

and the j^{th} entry of $\vec{a_k}$ is $a_{j,k}$. The same applies to the b variables. In practice, it is most useful to write the tableau with the Weyl operators stacked from left to right:

$$|\psi\rangle = \left[\begin{array}{ccc} \tau^{-\vec{a_1}\cdot\vec{b_1}} & \dots & \tau^{-\vec{a_{2n}}\cdot\vec{b_{2n}}} \\ \hline \\ a_{1,1} & \dots & a_{1,2n} \\ \dots & \dots & \dots \\ a_{n,1} & \dots & a_{n,2n} \\ \hline \\ b_{1,1} & \dots & b_{1,2n} \\ \dots & \dots & \dots \\ b_{n,1} & \dots & b_{n,2n} \end{array}\right]$$

Consequently, the blocks are stacked by phase, Z, and X, from top to bottom. Operating on this tableau via Clifford gates is nearly identical to the prime case in the sense that the conjugations are the same, but they now correspond to rewrite rules on *rows* of the tableau instead of columns.

C.3 Measurement in the non-prime dimension qudit stabilizer tableau

We wrote the tableau earlier in an odd fashion since this form, when regarded as a matrix A over $\mathbb{Z}_d$, allows us to reduce the problem of determining measurement outcomes by computing a normal form as in the prime case. However, $\mathbb{Z}_d$ is no longer guaranteed to be a field when d is an even, non-prime number, so we cannot use Gaussian elimination. Instead, we use the *Smith normal form* (SNF) of A that expresses it as USV, where U, V are invertible and S is a diagonal matrix.

A key feature of the procedure is that random Z measurements might not be uniformly random over all possible eigenvalues of Z. While the Weyl operators are the most general way to represent tableaus of arbitrary dimension, in practice they are incredibly expensive to compute with due to the practical inefficiencies of SNF algorithms on large matrices. Regrettably, the current build non-prime tableau simulation is often beaten by statevector simulation.