SANDIA REPORT

SAND2020-10518
Printed September, 2020

Sandia National Laboratories

Efficient, Scalable Tomography of Many-Qubit Quantum Processors

Erik Nielsen

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods

ABSTRACT

Quantum computing has the potential to realize powerful and revolutionary applications. A quantum computer can, in theory, solve certain problems exponetially faster than its classical counterparts. The current state of the art devices, however, are too small and noisy to practically realize this goal. An important tool for the advancement of quantum hardware, called *model-based characterization*, seeks to learn what types of noise are exhibited in a quantum processor. This technique, however, is notoriously difficult to scale up to even modest numbers of qubit, and has been limited to just 2 qubits until now. In this report, we present a novel method for performing model-based characterization, or tomography, on a many-qubit quantum proessor. We consider up to 10 qubits, but the technique is expected to scale to even larger systems.

CONTENTS

LIST OF FIGURES

SUMMARY

This report presents the research work of a 3-year LDRD project entitled “Efficient, Scalable Tomography of Many-Qubit Quantum Processors”. It describes the primary results coming out of the research, as well as some of the pitfalls and wrong paths taken along the way. The main product of this research is a prototype method to perform quantum tomography on many-qubit quantum processors and the theory behind it. We describe this many-qubit tomography after giving some context for how this problem fits within the field.

1. INTRODUCTION

Quantum computing is a relatively novel means of computation that prepares, manipulates, and ultimately measures a quantum mechanical state while maintaining its coherence - a fragile property that, once lost, cannot be recovered.[8] Often this state is spread across many effective 2-level systems called quantum bits, or *qubits*. In theory, a quantum information processor (QIP) could speed up certain problems by a factor exponential in the problem size relative to today's classical computers. However, achieving such impressive performance gains on problems of practical significance would require hundreds to millions of qubits, all operating with at very low error rates. Current quantum processors are small (2-50 qubits) and noisy (high error rates), and are unable to solve practical problems faster than their classical counterparts.[9] The path to a useful quantum processor – a chief goal of the field – then, is continued improvement of quantum hardware: the creation of quantum processors with more qubits and lower error rates.

One way of improving hardware is by using ad-hoc methods and intuition within the physics labs where the pioneer quantum processors are built. In small systems of up to few qubits, such in-lab intuition is a very useful and efficient way to proceed. Experimental teams' expertise in their own hardware leads to simple tests that probe and correct for noise in the systems. As the system becomes larger and more complex, however, it becomes increasingly difficult identify noise sources and intuit ways of combating them.

At this point, the methods of Quantum Characterization Verification and Validation (QCVV) become relevant. QCVV is an area of research that seeks to assess and understand the performance of QIPs largely from a hardware-agnostic standpoint. It accomplishes these goals through the use of *QCVV protocols*. A QCVV protocol analyzes the output of running a (usually pre-defined) set of circuits on the QIP with the purpose of better understanding the QIP's performance. This understanding takes different forms. QCVV protocols can be roughly divided into *holsitic methods*, which measure the overall performance of a device, and *tomographic methods*, which seek a detailed and predictive understanding of the noise.

Holistic methods are most helpful at succinctly measuring the capabilities and performance of existing devices. They may be used to guide the development of hardware, e.g. by measuring the performance of several different variants and choosing the best-performing one, but this usage doesn't typically provide any insight into what types of experimental modifications would increase the performance. As such, using holistic QCVV methods to navigate a path to improved hardware is similar to optimizing a function without any derivative information. These methods more useful for confirming, rather than driving, improved hardware performance.

Tomographic methods, on the other hand, are intended to map out noise in greater detail. We focus on one particular type of tomography we call *model-based characterization*. The goal of model-based characterization is to construct a predictive model of a quantum processor (or a

portion of it) based on *training data*. It attempts this by optimizing the parameters of a given statistical model so as to maximize the likelihood between the model and its training data. If a good fit cannot be obtained, the parameterized model must be enlarged or otherwise modified. Ideally, the model's parameters can be related to physical effects and/or controls in the laboratory. When this is true *and* a good fit is obtained, the resulting best-fit model provides significant insight into the error mechanisms at work and/or how to mitigate them. This knowledge then leads to improvements in the hardware, either by tuning the current device or through the design of next-generation devices. Thus, model-based characterization serves as an important tool in furthering the field of quantum computing.

Most QCVV protocols are limited in the number of qubits they can analyze at once. Holistic methods offer better scaling in this regard, and some state of the art approaches can be applied to 20-qubit processors without problem[11]. On the other hand, tomographic methods such as quantum process tomography and gate set tomography (GST) [6], are limited to 2-3 qubits – far below the size of today's processors. This LDRD has succeeded in extending tomographic methods to more qubits, and so furthered the state of the art in this area. In the remainder of this report, we describe our development of a tomographic protocol - a means of model-based characterization - that can be applied to up to O(10) qubits.

2. MODEL-BASED CHARACTERIZATION ON MANY QUBITS

In this section we describe a way to perform model-based characterization on processors with up to O(10) qubits. Our approach is based on concepts from GST, and in some ways can be seen as an extension of GST that can be applied to larger processors. Before discussing the distinguishing aspects of the many-qubit approach, we outline its basic structure and highlight its similarities with GST. The basic steps common to GST and the many-qubit tomography presented in this work are as follows:

1. **Select a model.** Model-based characterization ultimately optimizes a parameterized statistical model. The function of the model is to, for a given set of parameter values $\vec{\theta}$, map circuits into predicted outcome probabilities. Selecting a model means specifying exactly how this is done, including how many parameters the model has. GST models represent each QIP circuit layer (time step) as a dense matrix, and treat almost every element of these matrices as an independent parameter.

2. **Construct training circuits based on the ideal model.** The training circuits are selected so that their corresponding outcome probabilities (predicted by the model) are sensitive to changes in the model parameters. Moreover, we demand that this sensitivity increases linearly with the length (depth) of the circuits, so that small changes in the model parameters are *amplified* to become large changes in the outcome probabilities of long circuits. The careful selection and structure of the training circuits allows, in theory, Heisenberg-like scaling of the best-fit model's accuracy with respect to circuit depth. (Heisenberg scaling means the protocol makes efficient use of experimental resources.) Constructing a set of training circuits with these desired properties, however, is nontrivial. The circuits all have a particular structure, consisting of a *preparation fiducial* circuit followed by a *germ* circuit repeated some number of times followed by a *measurement fiducial* circuit. GST uses a three-part process to identify training circuits whereby 1) fiducial circuits are selected, then 2) germ circuits are selected, and finally 3) redundant fiducial pairs are eliminated on a per-germ basis.

3. **Fit the model to the training data.** The final step is to maximize the likelihood between the model and the training data. A multi-stage optimization method is used (see section 5) during which the likelihood function is repeatedly evaluated.

GST implements each of these steps using techniques that are practically limited to 1- and 2-qubit systems. Extending them to more qubits presents a number of challenges, which we note in turn below. These challenges are what make many-qubit tomography a hard problem, and why many of the accomplishments of this LDRD were thought to be intractable or at least not worth pursuing prior to this work.

- **The size of the model.** As the number of qubits increases, the number of parameters in models that seek to capture generic noise can increase very quickly. For example, the Markovian models used by GST have parameter counts that scale exponentially. This means that the optimization problem (a non-convex, minimization) rapidly becomes intractable. Moreover, even if the optimization could be performed, extracting meaningful results relevant to improving future hardware from such huge models would provide a challenge in and of itself.
- **Training data selection.** If we require the same Heisenberg-like accuracy scaling as GST, training circuits must be chosen to amplify changes in the model parameters. The somewhat ad-hoc three-part process used by GST cannot trivially be scaled up, suggesting that a completely new approach is needed. While it is tempting to simply relax the requirement of Heisenberg-like scaling, the structure of the GST circuits is also believed to help the model optimization avoid local minima, making the move to a less structured set of training circuits a risky prospect.
- **The size of the Hilbert space.** The most well known and appreciated impediment to performing tomography on many qubits is the sheer difficulty of simulating the outcomes of a (noisy) quantum circuit on many qubits. Standard techniques require time and memory resources proportional to 4^n for n qubits. While more efficient techniques are known for restricted sets of circuits, these restricted sets do not coincide with the noisy gates anticipated in real devices. Step 3 above requires the repeated evaluation of an objective function that itself requires the simulation of many quantum circuits. State of the art algorithms can perform *single* simulations on around 25 qubits using HPC systems (50 qubits using just pure state evolution[5]). Assuming a model with ≈ 1000 parameters, a training set of ≈ 1000 circuits, and a optimizer that requires ≈ 100 iterations (these are intended to be realistic numbers), performing tomography would require $10^8 > 4^{13}$ circuit simulations. This suggests that performing model-based characterization on around $25 - 13 = 12$ qubits would strain cutting edge HPC systems, and that around 8 qubits would be the most more modest HPC resources would allow.

In the sections that follow we describe how all of these challenges have been partially overcome by the work performed during this LDRD. In section 3, we describe a novel class of "cloud-noise" models whose parameters scale polynomially with number of qubits. Section 4 presents an efficient and scalable method for selecting training circuits that amply all the parameters of a cloud-noise model. Section 5 describes improvements to standard optimization techniques that allow us to successfully fit cloud-noise models to training data. We also show that data from actual quantum processors can be adequately described by a cloud-noise model, supporting our claim that these are *useful* and *relevant* low-parameter models. In section 6 a novel algorithm for circuit simulation is presented and tested. This algorithm was designed to overcome the Hilbert-space size constraints in situations when the overall amount of noise on the qubits is small, and we demonstrate its use on up to 10 qubits.

3. NOISE MODELS

Central to model-based characterization is the model itself. A model describes how some fixed number of real-valued parameters, $\vec{\theta}$, are used to construct the operations of the QIP under investigation. A model with more parameters can (typically) explain more intricate features of a data set, but requires more resources to optimize and can be more prone to over-fitting. GST uses a class of models in which every possible circuit layer is represented by a dense $4^n \times 4^n$ process matrix, where n is the number of qubits, whose elements constitute independent parameters. This is a natural choice, as it allows operations to be any time-independent "Markovian" process - but the number of parameters in this class of models obviously scales exponentially.

To keep the number of model parameters from increasing too quickly, QIP operations need to be represented in a *sparse* format, meaning the noisy operations must be specified by a small number of free parameters (many fewer than the number of elements in their superoperator representation, 4^{2n}). If we restrict ourselves to Markovian models (where each operation has a time-independent process matrix), then this sparsity means that not every type of Markovian error can be independently allowed. We hypothesize that wisely choosing which independent errors to include and which to exclude will result in a model that can predict real-device data despite having relatively few adjustable parameters. Restricting the allowed types of noise based on a given type of hardware too much will result in a model that, for better or worse, can only capture the noise from that type of hardware. Not restricting enough will result in a model with an intractably large number of parameters. We have developed a class of models that, we believe, achieves a good balance in this regard. The models we present are able to describe a wide range of physical noise sources and possess a number of parameters that scales linearly with the number of qubits. To describe this class of models, we introduce the language of error generators.

In what follows, we work in the framework of superoperators, so that gates on n qubits are $4^n \times 4^n$ matrices, states are 4^n-dimensional column vectors, and measurements are 4^n-dimensional row vectors. Gates are represented by symbols in normal italic font. Preparations and measurement outcomes are represented using super-kets, e.g., $|\rho\rangle\rangle$, and super-bras, e.g., $\langle\langle E|$, respectively. This allows us to write the evolution of a density matrix using notation parallel to the traditional quantum bra-ket notation for states. For example, $G|\rho\rangle\rangle$ is the density matrix resulting from preparing ρ and applying superoperator G, and

$$\langle\langle E_i|G|\rho\rangle\rangle \tag{3.1}$$

is the real-valued probability of obtaining outcome E_i after preparing ρ and applying G.

3.1. Error generators

The concept of error generators lets us neatly and powerfully talk about the errors on a gate operation. Let G be a n-qubit gate operation in question and $G^{(0)}$ be the corresponding ideal (perfect) operation. We separate the intended action of G from unintended "errors" by writing

$$G = \Lambda G^{(0)}, \tag{3.2}$$

which defines the *error map* Λ. Since $G^{(0)}$ is known and fixed as the ideal operation of the QIP, parameterizing G amounts to parameterizing Λ. Physics-based arguments (e.g., the form of the Schrodinger and Lindblad equations) motivate that we consider the *logarithm* of Λ, which we denote Γ, rather working with Λ directly. We assume this logarithm is well defined (it almost always is), and write:

$$G = e^{\Gamma} G^{(0)}. \tag{3.3}$$

We call Γ the *error generator* of G. We restrict it to having the Lindblad form,

$$\Gamma = \sum_{i=1}^{4^n-1} \alpha_i H_i + \sum_{i,j=1}^{4^n-1} \beta_{ij} S_{ij} \tag{3.4}$$

where α_i are real values, β_{ij} are complex values, and H_i and S_{ij} are maps given by

$$H_i: \quad \rho \quad \rightarrow i[P_i, \rho] \tag{3.5}$$

$$S_{ij}: \quad \rho \quad \rightarrow P_i \rho P_j^{\dagger} + \frac{1}{2}\left(\rho P_i P_j^{\dagger} + P_i P_j^{\dagger} \rho\right). \tag{3.6}$$

The P_i are Pauli matrices on n qubits, with P_0 the identity. The total number of Pauli matrices is 4^n, and so the sums in Eq. 3.4 have the P_i and P_j in Eqs. 3.5 and 3.6 ranging over all $4^n - 1$ non-identity Paulis. The H_i act as *Hamiltonian*-type errors (rotations) about the P_i Pauli axis, whereas the S_{ii} operator corresponds to P_i-type Pauli-stochastic errors (for 1 qubit, these shrink the Bloch sphere about the given Pauli axis). The off-diagonal S_{ij} operators serve to rotate the stochastic errors and to perform other types of errors (e.g. amplitude damping) that we do not discuss further here.[1] We refer to the H_i and S_{ij} collectively as *Pauli error terms*, and to α_i and β_{ij} as *Pauli error coefficients*. Notably, when β_{ij} is positive semi-definite, then Λ is a completely-positive trace-preserving (CPTP) map. The number of Pauli error coefficients in this case is $(4^n-1) + (4^n-1)^2 = (4^n-1)4^n$, which is, as expected, the same as the number of elements in a TP-constrained $4^n \times 4^n$ superoperator matrix (the TP constraint fixes the first row of the matrix).

The reason for introducing the error generator framework and writing G in terms of Pauli error terms and coefficients is that it provides a way of connecting physically intuitive types of errors with gate parameters. The errors associated with the H_i and S_{ii} are particularly intuitive, as these correspond to simple rotation and Stochastic errors on one or more qubits. Using error generators, we can easily restrict the types and the locality of Hamiltonian and Stochastic errors by setting some of the α_i and β_{ij} in Eq. 3.4 to zero. How we choose this "sparsity" is based on the following considerations.

- **Type.** As described above, Pauli error terms correspond to different types of errors, e.g. Hamiltonian, Pauli-stochastic, and other types. For example, we can restrict to models having just Hamiltonian and Pauli-stochastic noise by simply demanding that β_{ij} be a positive semidefinite diagonal matrix.
- **Weight.** The weight of a Pauli operator P is defined as the number of non-identity single-qubit Paulis present in the expression of P as the tensor product of single-qubit Paulis. For example, the 5-qubit Pauli $X \otimes I \otimes I \otimes X \otimes I$, written hereafter as $XIIXI$, has weight 2 whereas $IIYII$ has weight 1. The weight roughly corresponds to the number of qubit errors that happen simultaneously. Because high-weight errors are uncommon or precluded by nature (e.g. microscopic Hamiltonians typically have at most weight-2 terms), and because high-weight errors are problematic for quantum error correction, we consider models where the error weight is restricted to be at or below some maximal value. This maximal value is often 2 or 3, and is independent of n.
- **Locality.** Because errors often originate at a point or area in space, it is reasonable to expect that the errors on gates will exhibit some kind of locality. We define locality conditions by placing the qubits on a graph defining the nearest-neighbors of each qubit. Then, we restrict the Pauli error terms to those that have non-identity Paulis only within a *radius* of the gate's target qubits. The radius is specified as a number of nearest-neighbor hops allowed from (any of) the target qubits. For example, suppose we have 5 qubits arranged in a line. With a radius of 1, a gate that acts on qubit 3 is allowed to have Pauli error terms with non-identity Paulis on qubits 2, 3, and 4 only. For example, $IXXXI$, $IIXYI$, and $IZIII$ are allowed but $XIIII$ and $IIYIX$ are not. If the maximum weight allowed equals 2, then the $IXXXI$ term would be forbidden because it is weight-3.
- **Physical considerations.** A third means of constraining the number of error terms in Γ is based on the physics of the device itself. If, for instance, weight-2 ZZ-type errors are expected to occur but other errors are not, we could simply drop all weight-2 Pauli error terms except those containing two Z Paulis. A central motivation for using the Pauli error generator framework described above is that the physics underlying qubit devices is often neatly expressed in terms of the bit flip and phase errors corresponding to the Pauli basis elements.

3.2. Cloud-noise models

During this LDRD, we used the error generator framework given above to focus on a particular class of models we call *cloud-noise* models. The name comes from the model's ability to capture errors that occur in a "cloud" around a gate's target qubits, as depicted in Fig. 3-1. A cloud-noise model on n qubits is constructed as follows. The n-qubit idle operation is modeled as an independent gate, and is limited to at most weight-w_I Pauli error terms. There are no locality constraints (there is no notion of target qubits for a global idle gate), except that the up-to w_I nontrivial Paulis must correspond to a *connected* subgraph of the qubits. The error on each gate operation consists of a "local" part, followed by a "global" part. The local part is limited to Pauli error terms with weight at most the number of target qubits plus an "extra gate weight", w_g, that

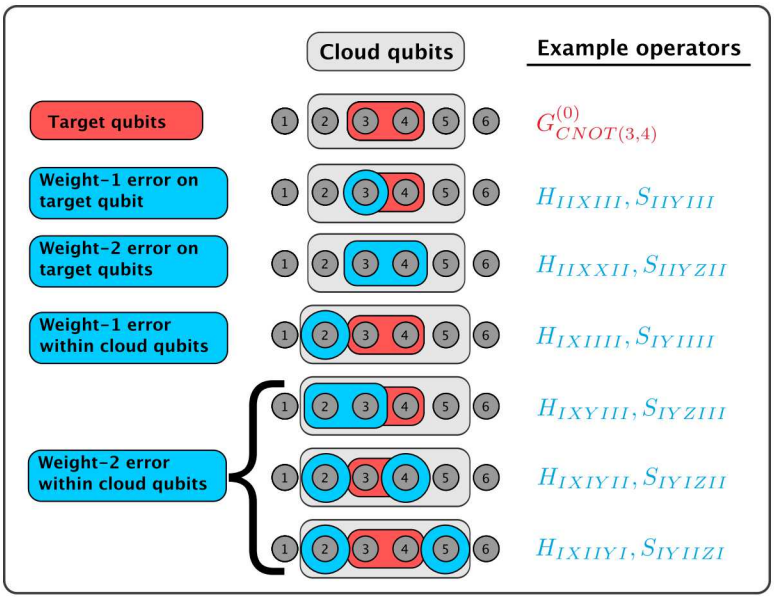

Figure 3-1. The types of errors, within a cloud-noise model, that are allowed on 2-qubit gate between target qubits 3 and 4. The entire device consists of 6 qubits arranged in a chain. This example is for $w_g = 0$ and $r_g = 1$, which means that a 2-qubit gate can have up to weight-2 ($(2+w_g) = 2$) errors on the 4-qubit "cloud" $\{2,3,4,5\}$ defined as those qubits reachable by $r_g = 1$ nearest-neighbor hops from any target qubit. The first line pictorially represents the ideal gate operating on its target qubits. The subsequent lines depict several different ways weight-1 or weight-2 errors can act on qubits within the cloud. Note that we do not demand that a weight-2 (or greater) error act on neighboring qubits. (If we added this constraint the final two lines would be disallowed.) The column on the right gives examples of some of the Pauli error terms that correspond to the picture to its left. The terms are written similarly to the H_i and S_{ij} of Eq. 3.4, except the Pauli P_i is used explicitly as the subscript (H_i is written as H_{P_i} above) and we write S_{P_i} in place of Eq. 3.4's S_{ii}.

occur within a radius of r_g (see Fig. 3-1 for an example). The global part is taken to be the same as the global idle gate. This, in effect, treats the global idle operation as an always-on background effect that we expect to align with the physics of many devices. We do not further reduce the terms in Γ based on physical considerations, as this would require interfacing with more detailed microscopic models of the device physics. We often, however, limit the types of the errors to just the Hamiltonian and Pauli-stochastic types. Sometimes we additionally include linear combinations of the off-diagonal S_{ij} error terms in order to allow for independent Pauli-affine errors (these map $\rho \to P$ for a Pauli P). We find that the number of parameters in such models scales linearly with the number of qubits, since increasing the number of qubits adds a constant number of gate operations, and each gate operation adds a constant number of error terms (i.e, parameters). Figure 3-2 shows how this scaling behaves for several choices of w_I, w_g, and r_g when the qubits are arranged in either a 1-dimensional chain or 2-dimensional square lattice. All the cloud-noise models we consider have $X_{\pi/2}$ and $Y_{\pi/2}$ gates on each qubit (single-qubit rotations by $\pi/2$ radians about the x- and y-axis of the Bloch sphere, respectively), and one controlled-NOT

(CNOT) gate between each nearest-neighbor pair of qubits. The models include only a single control-to-target direction for each CNOT gate (the other directions could be straightforwardly included as independent gates, but this would add to the parameter count).

In the numerical and experimental work detailed in section 5.2, we use models with $w_I = 2$, $w_g = 0$, and $r_g = 1$ on linear chains of 1-6 qubits, corresponding to the solid blue line in Fig. 3-2(a). We show there that such models can be used to fit simulated data, a nontrivial result given the complexity of the model and circuit selection (cf. section 4), and fit experimental data, supporting the desired result that cloud-noise models are able to capture the physics of actual quantum processors.

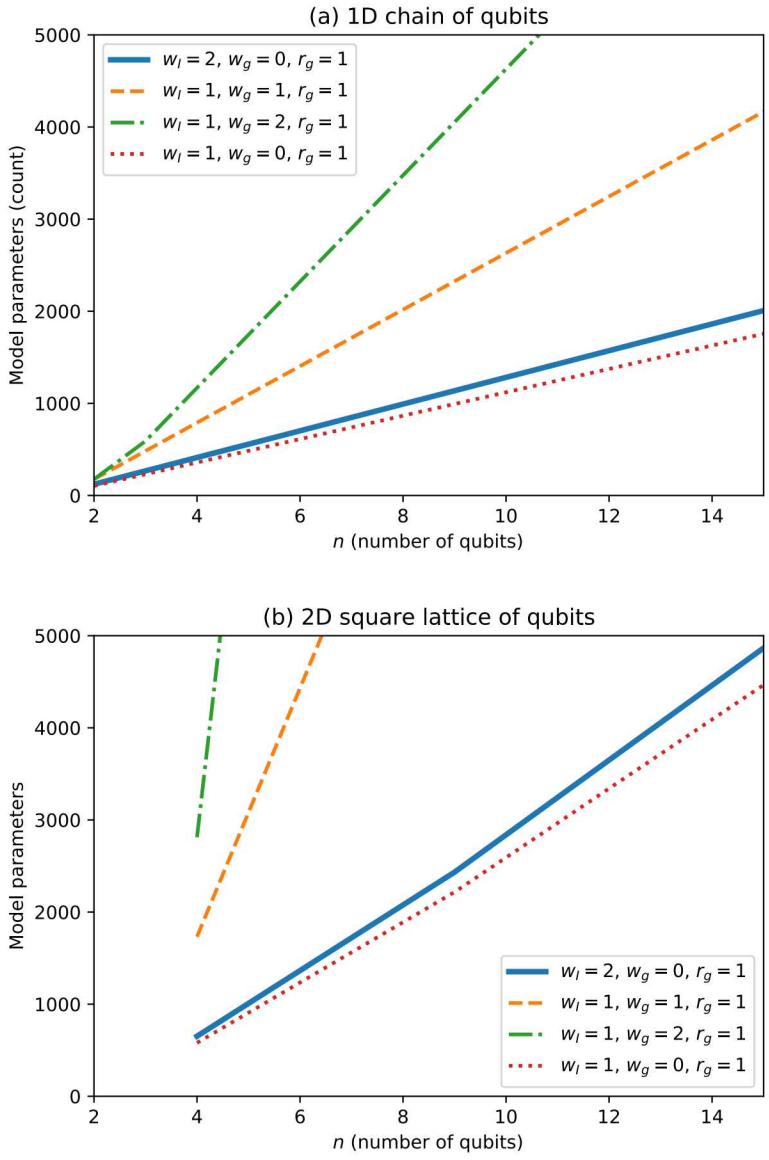

Figure 3-2. The parameter scaling of several classes of cloud noise models. The plots in (a) and (b) show the number of parameters in cloud noise models with $X_{\pi/2}$,$Y_{\pi/2}$, and uni-directional CNOT gates. Models are restricted to having only Hamiltonian and Pauli-stochastic errors, and the values of w_I, w_g and r_g shown in the captions. Upper and lower panes (a) and (b) show the scaling for n qubits arranged in a 1-dimensional chain or in a 2-dimension square lattice respectively. The case $w_I = 2$, $w_g = 0$, $r_g = 1$ (blue solid line) is particularly relevant, as this class of models is tested in section 5.

4. TRAINING CIRCUIT SELECTION

The second challenge to performing model-based characterization on many qubits is knowing what circuits are needed to probe the selected model – in this case a cloud-noise model. In this section we describe an approach, invented during the course of this LDRD, for selecting sets of circuits that are able to efficiently probe cloud-noise models. This means finding circuits that amplify all of a cloud-noise model's parameters.

In GST this problem is tackled by a 3-part process related to the structure of the circuits. GST circuits consist of a repeated germ circuit sitting between preparation and measurement fiducial circuits. The first part of GST circuit selection identifies fiducial circuits that generate complete sets of effective state preparations and measurements. The second and most computationally intensive part selects germ circuits. It repeatedly constructs Jacobian matrices containing the derivatives of the outcome probabilities of *all* the selected germ circuits with respect to *all* the model parameters. The size of this matrix, combined with the increased number of repetitions (trial germ sets) needed to construct a full germ set, results in this part scaling as a high-order polynomial with qubit number. The third part removes redundant pairs of fiducials. Redundancy exists because each germ circuit doesn't amplify every direction in its process matrix space. Overall, the poor scaling of the germ selection step requires we find a different approach to circuit selection, while the redundancy corrected for in the final step suggests that there is efficiency to be gained through alternative methods.

The scalable approach we present here selects circuits having the same germ-sandwiched-between-fiducials structure as traditional GST circuits. It takes as input the ideal model and a series of circuit lengths (depths) L_i for $i = 1 \ldots N_L$. These lengths dictate how many times the germs are repeated. The circuits are divided into N_L bins, one for each L_i. All of the circuits corresponding to germ g and the L_i-th bin contain g repeated $\lfloor L_i/l_g \rfloor$ times, where l_g is the length (depth) of g, sandwiched between some number of fiducial pairs. The L_i are usually logarithmically-spaced powers of 2. The purpose of the repeated germ is to amplify one or more linear combinations of the model parameters. The purpose of the pair of fiducial circuits is to expose the amplified quantities in the circuit outcome probabilities. This circuit structure is identical to that of standard GST.

The means of selecting germs and fiducial pairs in the present approach, however, is tailored to cloud-noise models (cf. section 3). The fact that parameters "belong" to different gates (including the global idle gate) allows us to focus on amplifying the parameters one chunk at a time rather than all at once. Gates' spatial locality (except for the global idle) and the limited weight of errors allow us to 1) restrict the search for germs to taking place on just a subset of the qubits, and 2) apply the results found for one subset of the qubits on another similar subset. The circuit selection process first finds circuits that amplify the parameters belonging to the global idle gate, and then circuits that amplify different "clouds" of gates (defined below).

4.1. Amplifying the global idle

Consider first how to amplify the parameters of the n-qubit global idle gate. By construction, this gate has errors with weight at most w_I, and so we can capture the effect of any individual error on a set of just w_I qubits. It is best not to think of these w_I qubits as any portion of the original device, but rather as a separate scratch space that we will use temporarily. Because the ideal idle gate commutes with all operations, any idle gate errors are amplified by simply repeating the idle gate itself. Thus, the germ-selection process is trivial for the idle gate: only one germ is needed, the global idle itself. All we need to do is find a set of fiducial circuit pairs that, when they sandwich the repeated idle, result in circuit outcome probabilities that together are sensitive to all of the idle gate's parameters.

This process is performed by iterating through a candidate set of fiducial pairs and checking whether the current pair makes any new (independent linear combinations of) idle gate parameters *accessible* as we define presently. Let the current fiducial pair under consideration be denoted (F_{prep}, F_{meas}), and denote the potentially noisy idle gate on w_I qubits as G_I. Using the super bra-ket notation described above, the E_i-outcome probability for the circuit formed by sandwiching the global idle, repeated m times, between the fiducial pair is:

$$p_i = \langle\langle E_i | F_{meas} (G_I)^m F_{prep} | \rho \rangle\rangle. \tag{4.1}$$

This equation assumes that the QIP has a single state preparation $|\rho\rangle\rangle$ and a single measurement that is represented as a positive operator-valued measure (POVM) with effects (outcomes) $\{E_i\}$. Using Eq. 3.3 to decompose G_I gives

$$p_i = \langle\langle E_i | F_{meas} \left(e^{\Gamma_I} I\right)^m F_{prep} | \rho \rangle\rangle, \tag{4.2}$$

where we have used the fact that $G_I^{(0)} = I$, the perfect identity. To consider the effect of small changes in the parameters (the α_i and β_{ij} in Γ_I, cf. Eq. 3.4), we can expand the exponential in a Taylor series,

$$\begin{aligned}
p_i &= \langle\langle E_i | F_{meas} \left((I + \Gamma_I + \ldots) I\right)^m F_{prep} | \rho \rangle\rangle && (4.3)\\
p_i &= \langle\langle E_i | F_{meas} \left(I + m\Gamma_I + \cdots\right) F_{prep} | \rho \rangle\rangle && (4.4)\\
p_i &= \langle\langle E_i | F_{meas} F_{prep} | \rho \rangle\rangle && (4.5)\\
&\quad + m \langle\langle E_i | F_{meas} \Gamma_I F_{prep} | \rho \rangle\rangle + \cdots . && (4.6)
\end{aligned}$$

After the simple rearranging performed in Eqs. 4.4-4.6, we find that the component of p_i that is linear in Γ_I and m, the number of germ repetitions, is simply $\langle\langle E_i | F_{meas} \Gamma_I F_{prep} | \rho \rangle\rangle$. This term can be obtained by computing the first-order polynomials for p_i when $m = 1$ and $m = 0$ and subtracting the two. The linear term, expressed as a function (a polynomial) of the parameters, can then be differentiated at the parameter-space point for the ideal model (usually $\vec{\theta} = 0$). This gives the *direction* in parameter space that is amplified by the circuit outcome probability p_i as m (the repetitions of G_I) is increased. Repeating this procedure for all of the outcomes (different $\langle\langle E_i|$) gives the set of amplified directions *accessed* by the given fiducial pair. We say that the fiducial pair accesses, rather than amplifies, certain directions since it is more properly the repeated germ that does the amplification.

If the idle gate on w_I qubits has N_I parameters, then we continue checking candidate fiducial pairs for new accessed directions until N_I independent directions are found. Since each fiducial pair represents up to $d-1$ independent circuit outcome probabilities, we expect to need a minimum of $N_I/(d-1)$ fiducial pairs. When this procedure finishes, a set of fiducial pairs for the G_I germ that access all of the (up-to-weight-w_I) errors on our w_I "scratch" qubits has been found.

The errors on the w_I scratch qubits may occur on *any* w_I-qubit connected subset of the n total qubits. Note that the one and only germ, the idle gate, naturally extends from w_I to n qubits. To ensure the outcome probabilities are sensitive to all up-to-weight-w_I errors on n qubits, we need a set of fiducials on n qubits such that their restriction to any connected w_I-qubit subset yields at a minimum the fiducial pairs found for w_I qubits using the procedure outlined above. This is closely related to the k-coverage problem (with $k = w_I$), and can be solved efficiently. We refer to this process as "tiling" the fiducials, as it extends the fiducials from w_I to n qubits in a way that bears some resemblance to repeating the fiducials on a subset of the qubits to all of the qubits. Tiling results in a set of n-qubit fiducials that access all the parameters of the global idle gate. Suppose there are N_A such pairs and denote them $\{F^{(i)}_{prep}, F^{(i)}_{meas}\}$ for $i = 1 \ldots N_A$.

Recall that these fiducial pairs all belong to the sole global idle germ. Since the global idle has unit length (depth) the final circuits for the i-th bin, corresponding to length L_i, are simply $\{F^{(j)}_{meas}(G_I)^{L_i}F^{(j)}_{prep}\}$ where $j = 1 \ldots N_A$.

4.2. Amplifying gate clouds

Next, we move on to find circuits that amplify the parameters of other (non-idle) gates. We divide up the gates into *clouds*, defined as a group of gates having the same target qubits and having the same neighborhood for potential errors as specified by w_g and r_g. For instance, all the single qubit gates operating on a given qubit would form a cloud, assuming that their maximum-error-weight $(1 + w_g)$ and radius (r_g) values were the same.

The purpose of clouds is to divide the circuit selection process into separable units. Let's consider a particular cloud. The gates within this cloud are different from the global idle gate in that 1) these gates' nontrivial action is restricted to a *local* neighborhood of qubits, call it S, around the cloud's target qubits, and 2) the ideal implementation of these gates is, in general, nontrivial on the target qubits. Point 1) means that we can restrict our search for germs and fiducial pairs to qubits in S, similarly to what we did with the group of w_I scratch qubits for the idle gate. A consequence of 2) is that germ selection is more involved, since naively repeating a nontrivial gate may "echo away" certain of its errors, i.e., some errors are made to cancel themselves via the action of the gate. Similarly to how we focused only on amplifying the parameters belonging the global idle gate above, here we restrict ourselves to just the parameters belonging to the gates in the cloud, which we refer to as the *cloud's parameters*. We repeat the following steps to find the circuits (germs and fiducial pairs) that amplify these parameters:

1. Take the next germ g for consideration from a candidate list. g can be a single gate, but in general is a short sequence of gates. The list of candidate germs is supplied as input, and only needs to contain circuits that act on the *target* qubits of the current gate cloud.

Non-target qubits are always idling in the ideal (error free) versions of the circuits under consideration, so errors on non-target qubits can be amplified using just the idle germ as in the global idle case. We take as our list of candidate germs all possible circuits acting on the target qubits that have length below a certain cutoff, and random longer circuits (this choice is not unique).

2. Compute the number of repetitions l such that the ideal g^l is the identity. We call g^l a *synthetic idle*. In all the QIPs we have considered, the native gates all have the property that such an l exists, and a synthetic idle can be found.

3. Find the parameter-space directions amplified by the synthetic idle g^l. This is done by finding a set of fiducial pairs for the germ g^l that access as many amplified linear combinations of cloud's parameters as possible. The procedure is identical to that used to identify fiducial pairs for the global idle gate with one important exception: We don't expect that repeating g^l will necessarily amplify *all* of of the cloud's parameters because g may echo away some errors. In this context, the fiducial-pair selection process stops only after we have tried every candidate pair. This is acceptable, since typically there are not that many candidates. The possibility of a germ being insensitive to some types of errors is why more than a single germ is needed, and why we must keep selecting germs until all the cloud's parameters are amplified.

4. If g^l amplifies new (independent) cloud-parameter directions, then g is accepted as a new germ. From the analysis of g^l in the last step we know which parameter-space directions repeating g will amplify (or, more properly, the *space* of amplified directions). However, when repeating g a number of times, b, that is not a multiple of l, *different* fiducial pairs will be needed to access these amplified parameter-space directions. This is in contrast to the case of the identity germ, where same fiducial pairs can be used at all repetition-counts. For a non-identity germ, the non-trivial action of the gate "twists" about the outcome probabilities so that to access the same known-to-be-amplified linear combinations of parameters in the circuit outcome probabilities, *different* fiducial pairs are needed. We can find these fiducial pairs by applying the same procedure we used for the global idle gate or g^l to the non-idle g^b gate with a simple modification. When considering a fiducial pair, instead of adding any pair that accesses one or more new parameter-space directions, we only accept pairs that access the particular directions found from the analysis of g^l, i.e., those that we know are amplified by g. In this way we can find, for each length L_i and $b = \lfloor L_i/l_g \rfloor$, a set of fiducial pairs to surround g^b such that all the newly amplified cloud-parameter directions are evidenced in the circuit outcome probabilities. Circuits composed of g^b sandwiched between each of the so-found fiducial pairs, are added to the i-th bin of circuits for the current cloud.

5. Otherwise, we reject the candidate germ and return to step 1 immediately.

6. Test if the number of independent amplified directions equals the cloud's number of parameters. If so, then the selection process ends. Otherwise go to step 1.

Our implementation (in pyGSTi[7]) groups clouds that differ only by qubit relabeling together into a *cloudbank*. The process above is repeated for each cloudbank (effectively, for each *distinct*

cloud). Grouping the clouds into cloudbanks avoids duplicate work, but more importantly allows straightforward and efficient tiling of the circuits on clouds with disjoint qubits. For example, consider a 6-qubit linear chain with single-qubit $X_{\pi/2}$ and $Y_{\pi/2}$ gates on each qubit. If each gate has an error radius of 1, then, e.g., the gates on qubit 2 can have errors on qubits 1, 2 and/or 3. Similarly errors on gates targeting qubit 3 and 4 are constrained to the qubit subsets {2,3,4} and {4,5,6}, respectively. The circuits for each of these three clouds have the same structure, and thus belong to a single cloudbank. When creating the final circuits, the circuits on the {1,2,3} and {4,5,6} clouds can be performed in parallel ("tiled") since the clouds are disjoint. Circuits on the {2,3,4} cloud would be performed separately (possibly with clouds from another bank, e.g. {5,6}, though this is not part of our current implementation). We tile the circuits found for the individual clouds in this way, one bank at a time. This is done separately for each bin of circuits (corresponding to the different L_i), resulting in final circuit sets for each bin. These circuits add to those found for the global idle to form a complete set of circuits that amplify all the parameters of the cloud-noise model.

5. FITTING THE MODEL TO DATA

Cloud-noise models are designed to capture common types of noise in many-qubit processors, and in the previous section we developed a procedure for finding circuits that are sensitive to such models' parameters. The next and final step in our approach to many-qubit tomography is to optimize over the parameter space of a chosen model, $\mathscr{M}$, resulting in an optimal set of model parameters. We refer to the 0-parameter model obtained by evaluating $\mathscr{M}$ at the best-fit point as the *best-fit model*. When our tomography protocol is successful, the best-fit model is a predictive model that describes the behavior of the device beyond just the training data.

The best-fit model, which we denote $\mathscr{M}_{\mathrm{MLE}}$, is found by maximizing the logarithm of the likelihood between the data and the model. The loglikelihood function being optimized is given by

$$\log \mathscr{L}(\vec{\theta}) = \sum_i n_i \log p_i(\vec{\theta}), \tag{5.1}$$

where i ranges over all the outcomes for all the circuits, n_i is the number of times the i-th outcome is observed, and p_i is the probability of that circuit outcome, as it is predicted by $\mathscr{M}$. Since the probabilities $p_i(\vec{\theta})$ are general, nonconvex functions, $\log \mathscr{L}$ is not convex. This precludes the use of efficient global optimization methods. We must instead resort to using local optimization methods that cannot guarantee a global optimum will be found. In spite of this, we have developed a heuristic optimization method that, we find, is able to nearly always find the global optimum in practice. We describe this method presently.

5.1. Optimization method

There are many out-of-the-box local optimizers that can be directly applied to optimize Eq. 5.1. For example, the `scipy` library contains a wide selection of general optimization methods. We find, however, that standard implementations succeed only some of the time, and often yield significantly non-optimal results. Over the course of this LDRD, many additions and updates were made to our optimization procedure, which began as the algorithm used for standard GST. What we present here represents the culmination of that work. We attribute our final approach's robustness to two important elements:

1. The optimization is performed in stages.

2. The local optimization method performed at each stage is robust to canyons and valleys, and less likely to attempt large steps.

As described in section 4, we bin the circuits according to length (L_i-value). The purpose of this segregation is so that the optimization can be performed in stages. The first stage optimizes $\log \mathscr{L}$ using only the circuits from the first bin (i in Eq. 5.1 ranges only over the circuit outcomes of circuits in the L_1 bin). The second stage uses the circuits from the first two bins, and so on, until N_L optimization stages have been completed. The stages are "daisy-chained", meaning that the best-fit found for stage $(k-1)$-th stage is the initial seed for stage k. When we take the $\{L_i\}_{i=1}^{N_L}$ to be powers of 2, each successive stages adds circuits that are roughly twice as long as the circuits of the prior stage. This gradual lengthening is intended to avoid local minima by providing a good seed to each stage of the optimization. This is crucial because the local optimization method that performs each stage can only be expected to minimize the objective function within a *basin*. The progression of stages tries to ensure that each stage is always seeded at a point in same basin as the global optimum. Since the outcomes of longer circuits are more sensitive to model parameters, adding longer circuits to the optimization makes more, smaller, and steeper basins. This creates a landscape with a sharper global minimum but also with more local minima, similar to the situation in standard GST[6]. In practice, when there are $N = 10^3$ - 10^4 samples per circuit, we find that taking the $\{L_i\}_{i=1}^{N_L}$ to be powers of 2 is sufficient to ensure the seed of each stage is in the correct basin. Failure is often detected by observing an extremely bad best-fit that gets increasingly bad with successive stages.

We perform the optimization at each state using a modified Levenberg-Marquardt (LM) method. The LM method solves a nonlinear least squares problem, and the log-likelihood can easily be transformed to have this form.[2, 6] It is an iterative method that chooses each step by interpolating between a Gauss-Newton step and a gradient-descent step. If $\vec{f}$ is the vector-valued least-squares objective function and J is its Jacobian matrix, then the step $\vec{\delta}\theta$ is computed by solving

$$\left(J^T J + \lambda D\right) \vec{\delta}\theta = J^T \vec{f}, \tag{5.2}$$

where λ is the *damping parameter* and D is the *scaling matrix*. A precursor to the LM method, introduced by Levenberg, set $D = I$ (the identity matrix). In this case, varying λ between 0 and ∞ clearly interpolates between the Gauss-Newton and gradient descent steps, respectively. The traditional LM method uses $D = \text{diag}\left(J^T J\right)$, the matrix formed by extracting just the diagonal of $J^T J$. The value of λ is adaptively updated at each iteration, and there are multiple suggested ways of performing this update [12]. We find that within the realm of suggested λ-update strategies, performance does not substantially depend on the precise strategy chosen. Choice of the scaling matrix, on the other hand, we find to change the optimization's behavior significantly, and so we focus on modifications here.

The purpose of the scaling matrix is to inflate or deflate the "global" damping amount λ applied to different parameter-space directions based on the curvature of the objective function in that direction. A large damping results in a step close to a pure gradient-descent step, and is often advantageous when the objective function is far from its optimum and/or on a plateau above a canyon. A low damping results in nearly Gauss-Newton step, which is often advantageous when near the minimum or when following a canyon. Levenberg's $D = I$ suggestion is called "additive damping" because λ is simply added to each coordinate direction, whereas the traditional LM method uses so-called "multiplicative damping", where the damping along each coordinate direction is proportional to the approximate curvature along that direction.

We modify the LM method in two significant ways, both centered around the scaling matrix:

- **The scaling matrix is placed in the basis of singular values of $J^T J$.** We find that the diagonal values of $J^T J$ are often bad proxies for the singular values of the matrix, and lead to poor scaling choices because the "principle axes" of the objective function's landscape (e.g. the direction of a canyon) do not lie along the coordinate directions. By evaluating the left-hand side of Eq. 5.2 within the basis of $J^T J$'s singular vectors we choose effective coordinate directions that better align with the objective function, and hence get a more accurate idea of which parameter directions are important.

- **The scaling matrix values are adaptive.** The scaling matrix is a diagonal matrix of the singular values of $J^T J$ raised to the power s (usually near 1.0). Our algorithm keeps track of the current value of s, which we denote s_0, and adapts it after every iteration, much like λ. At each iteration, three different $\vec{\delta\theta}$ values are considered: one using $s = s_0$; the other two using $s = s_0 \pm \delta s$, where δs is a small fixed number (we use 0.1). If the best decrease in $||f||$ is given by $s = s_0 + \delta s$, then this becomes the s_0 on the next iteration, and similarly for $s_0 - \delta s$. Note that when $s = 1$ and $J^T J$ happens to be diagonal, then $D = \text{diag}\left(J^T J\right)$, the same update as the traditional LM method. When $s = 0$, it results in $D = I$.

We also have experimented with allowing uphill steps and geodesic acceleration[12]. Allowing uphill steps can notably improve performance by reducing the number of LM iterations needed. This suggests that our landscape may have winding canyons that are helped by an ability to glide around a turn like a bobsled. We did not find any improvement using geodesic acceleration, and so disabled this feature in our final optimization algorithm. Overall, we find that our modifications to the scaling matrix have the most significant positive effect on algorithm performance.

5.2. Numerical validation

As stated earlier, the optimization problem at hand is not convex. This means that no efficient techniques are known to exist that globally optimize $\log \mathscr{L}$. The staged optimization algorithm described in the last section, while it attempts to mitigate the weakness of other local optimization methods, is not guaranteed to find a global optimum. In this section, we show that in numerical tests our proposed algorithm *is* able to find a global optimum. These results do not prove the method will always succeed, but they do establish that there are at least some cases when it does. Furthermore, they suggest that the method will succeed in cases similar to the physically-motivated tests.

For each fixed number of qubits, we assess the model-based characterization protocol described thus far by seeing whether a "base" parameterized model $\mathscr{M}(\vec{\theta})$ to able to fit training data simulated from a *true model* that is known to be describable by the base model. Our base model is a cloud noise models with Hamiltonian and Pauli-stochastic errors, and $w_I = 2$, $w_g = 0$, and $r_g = 1$. The true model $\mathscr{M}_{true} = \mathscr{M}(\vec{\theta}_{true})$ is created by fixing the parameters of the base model in a way that selects realistic random errors on the qubits as follows. For each gate, including the global idle, and for each state preparation and measurement operation,

- The coefficients of 3 randomly chosen weight-1 Hamiltonian errors (3 randomly chosen α_i in the operation's error generator) and 2 randomly chosen weight-2 Hamiltonian errors are set to values uniformly sampled from $[-0.0005, 0.0005]$.
- The coefficients of 3 randomly chosen weight-1 Pauli-stochastic errors (3 randomly chosen β_{ii} in the operation's error generator) and 2 randomly chosen weight-2 Pauli-stochastic errors are set to values uniformly sampled from $[0, 0.0005]$.

These errors mimic an expected situation in which a small subset of the error terms dominate the others. In addition to these errors, a uniformly random value from the interval $[0, 10^{-5}]$ is added to each α_i and β_{ii} coefficient of the global idle gate. Since the global idle operation is applied at the end of every circuit layer, these random errors mimic the presence of low-level background noise that is not in any particular direction.

After $\mathscr{M}_{true}$ is created, training data are produced by, for each training circuit, sampling N times from the multinomial distribution constructed from the circuit's outcome probabilities. Recall that the training circuits are chosen to amplify movement along all directions within $\mathscr{M}$'s parameter space. Since the true model corresponds to a point within the parameter space of the model being optimized ($\vec{\theta}_{true}$), we expect the tomography to succeed in reconstructing $\mathscr{M}_{true}$ from the training data. When there is no finite sample error (the $N \to \infty$ limit), this reconstruction should be exact, and we have verified that this is the case on 1-6 qubits. When there is finite sample error, the best-fit model, $\mathscr{M}_{\mathrm{MLE}} = \mathscr{M}(\vec{\theta}_{\mathrm{MLE}})$, should fit the data slightly better than the true model since it is able to fit some of the statistical noise.

To quantify this, we measure the obtained versus expected goodness-of-fit by applying Wicks' theorem. Wick's theorem states that, given two *valid* models $\mathscr{M}_1$ and $\mathscr{M}_2$, twice their log-likelihood ratio, $\lambda = 2(\log \mathscr{L}_1 - \log \mathscr{L}_2)$ should be χ^2_k-distributed, where $k = N_1 - N_2$ is the difference in the models' number of parameters. We define the number of standard deviations λ is above it's mean k as

$$N_\sigma = \frac{\lambda - k}{\sqrt{2k}}. \tag{5.3}$$

When $\mathscr{M}_1$ is known to be valid, $N_\sigma \gg 1$ is strong evidence that $\mathscr{M}_2$ has been violated, and that a larger (or different) model is needed to explain the data.

The points in Fig. 5-1 give N_σ from comparing the best-fit estimate from data with $N = 10,000$ with the true model and with the so-called "maximal model". The maximal model, by definition, contains one parameter for every degree of freedom in the data, and therefore fits the data perfectly. The number of degrees of freedom in the data is roughly equal to the number of independent counts but, we find, must be corrected for small-N effects. This is especially true as the number of qubits gets large, since small circuit outcome probabilities result in many outcomes having few observed counts. The maximal model is a useful reference point, as it can be computed directly from the data without knowledge of the true model. As a consistency check, Fig. 5-1 also compares the maximal and true models, both of which are expected to always be true by construction.

We find that our tomographic protocol successfully finds a good fit ($N_\sigma \leq 3$ between the estimate and true models) for 1-6 qubits. Comparison with the maximal model also give low N_σ values as

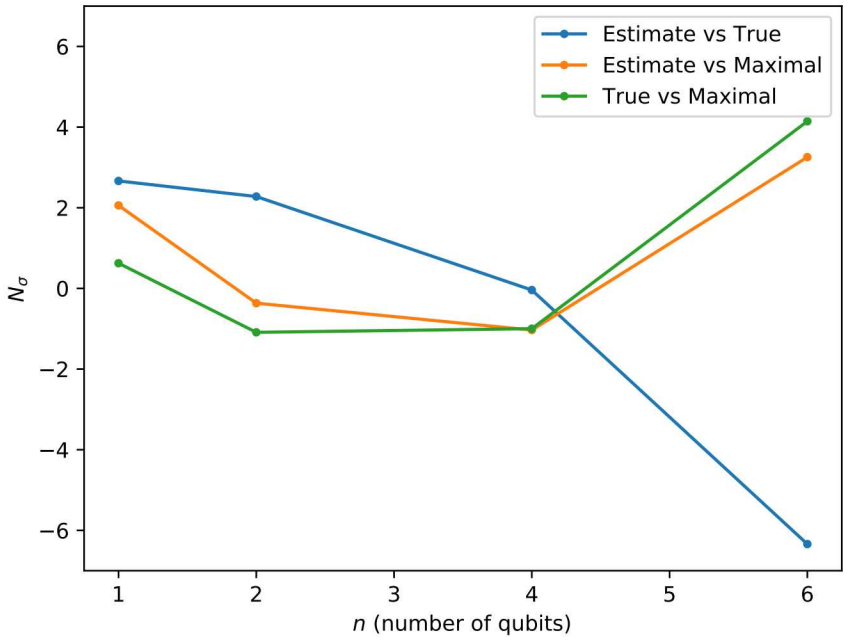

Figure 5-1. Validation that multi-qubit tomography is able to reconstruct an underlying "true" model on 1-6 qubits. For each qubit number, a cloud-noise model is created with random noise as described in the text. Data generated from this model, with $N = 10,000$ samples per circuit, are fit to a cloud-noise model with the same structure (Hamiltonian and Pauli-stochastic errors, $w_I = 2$, $w_g = 0$, and $r_g = 1$) but that initially has no errors. Wilks' theorem is applied between pairs of models as indicated in the legend. Low N_σ indicates that the first model listed in the legend (the best-fit estimate or true model) is valid. In the comparison between the best-fit and true model (green line), low $N_\sigma \leq 3$ indicates that the tomography has reconstructed the true model as well as can be expected. Comparisons between each model and the "maximal model" described in the text (blue and yellow lines) indicate that estimate and true model explain the data, but with a slight uptick at the $n = 6$ that we believe is from an inaccurate calculation of the maximal model's degrees of freedom (see text for details).

we expect, but with a notable up-tick for 6-qubits. We believe this is because finding the exact number of degrees of freedom in the maximal model is complicated by low-N effects, and that our accounting for this is slightly inaccurate at $n = 6$. Overall, these results suggest (but do not prove) that our iterative local optimization method, paired with our approach to circuit selection, is able to successfully navigate to a global optimum. We do not observe any cases where the optimizer clearly gets stuck in a local optimum. This is in part due to improvements made to the optimization method (see section 5.1) and in part, we suspect, because we consider small noise magnitudes and a fairly large N (sample count). We expect that, similar to GST, the optimization will fail when N drops below a critical value and/or the errors are large. A more comprehensive study of conditions for which the optimization method fails is a topic for further work.

5.3. Experimental validation

In addition to being able to handle the challenges of the global optimization problem, successful model-based characterization must begin with a model rich enough to describe actual noise present in devices. So we ask: *Are cloud-noise models able to describe realistic noise?* Whereas the global optimization problem lies within the realm of computer science, the answer to this question is governed by physics, and is difficult to answer absolutely. At the time of this writing there are many QIPs in existence. They are based on multiple material systems (largely superconductors and trapped ions), and this diversity is expected to grow in the future. Thus, we expect a wide range of physical effects to be included in "realistic noise". Still, a critical step in the evaluation of any model-based characterization must be to test the protocol on existing devices, and in so doing answer this foundational question as best as we can. Since cloud-noise models were designed to capture fairly general noise, we have at the outset a hope and expectation that they will be able to describe a wide range of physically relevant noise.

We test how well our model-based characterization approach works on a number of extant quantum processors. For each tested processor we 1) select an appropriate cloud-noise model $\mathscr{M}$ (i.e., choosing the types of errors to allow, locality, etc.), 2) generate training circuits based on the ideal $\mathscr{M}$ using the method of section 4 and execute them on the QIP, and 3) optimize $\mathscr{M}$'s parameters so that the model's predicted probabilities best fit the training data. We again use N_σ (cf. Eq. 5.3) to quantify how well the model fits the data, now having no choice but to compare the estimated model with the maximal model. Ideally, the best-fit model would be able to explain all of the data, as in the simulated-data case, so $N_\sigma < 2$.

Figure 5-2 shows N_σ obtained from applying our protocol to actual quantum processors. These QIPs range in size from 2-8 qubits, and all have superconducting transmon qubits. (Ion trap processors were not tested due to the lack of public availability of larger processors.) The processors we tested came almost entirely from Rigetti Quantum Computing (RQC), with one exception being a device fabricated by IBM and run by BBN Raytheon. In each case, we use a cloud noise model that is restricted to Hamiltonian, Pauli-stochastic, and Pauli-affine errors only, and with $w_I = 2$, $w_g = 0$, and $r_g = 1$. Affine-type errors were specifically included because superconducting qubits are known to suffer from "T1 decay", an error process whereby the $|1\rangle$

state decays into the $|0\rangle$ state (but not vice versa) after some time. Affine errors are required, in tandem with stochastic errors, to model amplitude damping.

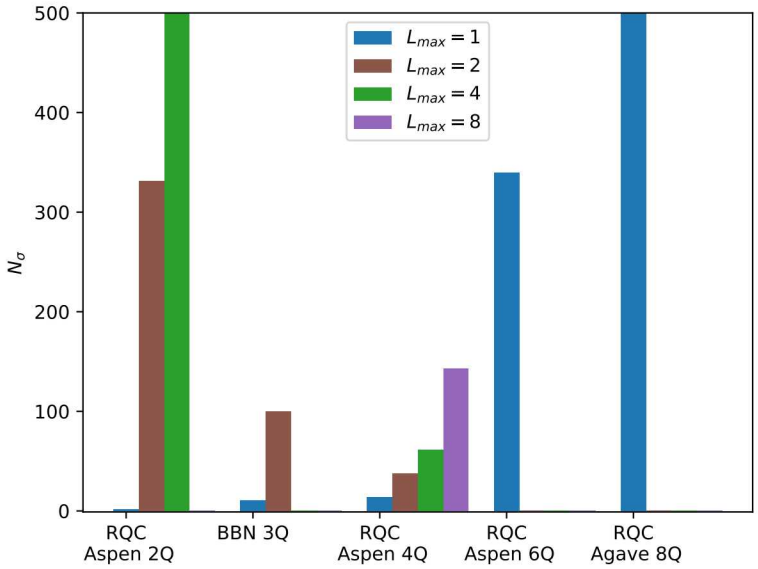

Figure 5-2. Results of fitting cloud-noise models to data from real quantum devices. Our tomographic approach was applied to five superconducting-qubit devices. Four were Rigetti Quantum Computing devices (three used their "Aspen" hardware and and one their "Agave") and one IBM device run by BBN Raytheon. How well a cloud noise model (of the class tested) is able to describe the data is measured by the number of standard deviations, N_σ of model violation we observe. Higher values of N_σ indicate more certainty that the (best-fit) model is invalid, i.e. that it doesn't describe all of the data. Different color bars show N_σ for when fitting data from different sets of circuits. The circuits use to perform the tomography are binned according to the maximum length of their germ-power, L. N_σ values for a given L_{max} include all the circuits with $L \leq L_{max}$. Absent bars indicate optimizations that were never run, e.g. only $L = 1$ circuits were used to analyze the 6- and 8-qubit devices. We find that even shallow ($L = 1$) circuits on the 6- and 8-qubit devices cannot be described by a cloud-noise model of the type we tried. The other three devices show near-consistency (low-N_σ) for shallow circuits but that the certainty of model violation grows with increasing L. This behavior is typical when the model is able to capture much but not all of the device behavior.

The results presented in Fig. 5-2 show mixed success. In the largest devices - 8-qubit Agave and 6-qubit Aspen - the cloud noise model is clearly unable to describe the data. Even the initial ($L_{max} = 1$) optimization stage was unable to produce an adequate fit, causing us to stop the test there. In the other, smaller, devices, the model is able to describe the data from shallow circuits ($L_{max} = 1$ and sometimes $L_{max} = 2$) but not from deep circuits. We see, particularly in the Aspen 4Q device, a step-like behavior of N_σ that is typical when the model describes the QIP's behavior moderately well. Here the data from shallow circuits are consistent with model predictions but longer (deeper) circuits, which provide more sensitive tests of model violation, yield data sets that

indicate increasing certainty of model violation.

While this performance may initially appear disappointing, we conclude that these cloud-noise models' ability to capture real device noise is promising. We take this optimistic view because there is reason to suspect that the noise in the larger devices we tested is substantially higher, even on a per-qubit basis, than that in smaller devices. By construction, the cloud-noise models describe fewer kinds of errors than full-blown GST (this is what avoids having their number of parameters grow too quickly). However, running single or two-qubit GST on subsets of the larger devices reveals that these devices posess noise that not even GST's Markovian error model can capture. This indicates that the dominant sources of noise are *not* Markovian single or two-qubit errors. We expected a scenario where the errors on a multi-qubit device would be similar to those of isolated qubits (or pairs of qubits), with some additional error coming from couplings with the other qubits. The errors in larger devices seem, instead, to be dominated by effects that are either high-weight (greater than weight 2) and/or non-Markovian. The assumptions that went into our cloud-noise models (low-weight, locality, Pauli-alignment) are based on a physical picture where high-weight errors are perturbations on local errors. Devices that cannot even remotely be explained by local Markovian models (1- and 2-qubit GST models), such as the 8-qubit Agave and 6-qubit Aspen devices, we do not expect to be describable by cloud-noise models either. We expect that as experimental groups target such non-Markovian effects as drift and heating, which can be identified by separate means,[10] we will see 5-10 qubit devices with noise similar to the smaller devices and that are describable by cloud noise models.

When we restrict ourselves to the smaller 3-4 qubit devices, the chosen cloud-noise models are able to describe shallow but not deep circuit data. This indicates that the model is able to roughly reproduce the data, but not well enough to pass the demands of statistical validity. We expect in these situations that although the model is certainly invalid, a small amount of "slack" can be added to the predicted circuit probabilities to achieve statistical consistency with the data. This "wildcard error" is the topic of an ongoing and parallel research effort, and combining it with the approach developed here is a topic of further study. Thus, we expect that only minor tweaks to the model or improvements to the hardware (e.g., to combat drift or other time-dependent effects), will allow cloud-noise models to accurately predict these smaller processors' execution of deeper circuits.

6. CIRCUIT SIMULATION VIA PATH INTEGRATION

Evaluating the log-likelihood objective function (Eq. 5.1) requires computing the model's predicted probability $p_i(\vec{\theta})$ for each observed circuit outcome i. We therefore require strong simulation of circuit outcome probabilities, as the probability distributions themselves are needed - sampling from them alone (weak simulation) is insufficient. In the results presented so far we have computed outcome probabilities by propagating a density matrix forward in time, as implemented in `pygsti`[7]. Over the course of the LDRD, the circuit simulation capabilities in `pygsti` were updated to become more time and memory efficient. We achieved this primarily by eliminating unnecessary copying and handling a significant amount of the computation in more efficient C-code. Density matrix propagation requires storing a real vector (the density matrix) of size 4^n and repeatedly multiplying it by sparse matrices - a task with exponential resource scaling in n. We find that performing cloud-noise model fitting beyond 6-8 qubits becomes intractable using modest HPC resources (100s of processors and 10s of gigabytes of memory per processor). A more streamlined HPC code could probably eek out another 2 or 3 qubits, but with substantial development cost and CPU time.

An alternative would be to repeatedly sample from the outcome probability distribution (weak simulation) until a convergence tolerance is reached. This approach has the advantage that it scales as 2^n times the number of samples needed for convergence, which is often less than 4^n. A downside to this approach is that it becomes difficult (expensive) to compute accurate derivatives of the probabilities, which are helpful to optimization approaches in general and essential to the particular LM approach we presented above. It also fails to remove the exponential scaling of circuit simulation with n. Still, this approach has the merits of a yielding a clear reduction in resources and having a straightforward implementation, and we would like to see it pursued in future work.

In this work we took a different tack, and explored an alternative approach that circumvents, in certain regimes, the exponential scaling of traditional circuit simulation. This approach has allowed us to extend our model-based characterization to systems of more than 10 qubits.

The core idea is to replace matrix multiplication, which has exponential-in-space scaling ("space" here is the number of qubits, i.e., size of the processor) with an exponential-in-time path integral. By intelligent pruning of the paths, imposing suitable conditions on the magnitude of the errors (that they be small), and limiting the length (duration) of the circuits, we are able to mitigate the exponential-in-time scaling and push the tomography protocol to handle higher numbers of qubits.

In the end, we observe that the technique provides a dramatic speedup in the regime where circuits are short (low depth) and errors are small. The shallow-depth, small-error regime is of particular interest because only shallow-depth circuits are needed for tomography (consider

process tomography), and there may be less need to characterize many qubits at once when some or all of the qubits have large errors. In large-depth or high-error regimes, where the method is not intended, it will become intractable even for fewer than the 6-8 qubits that can be performed using traditional strong simulation methods.

We can write a circuit outcome probability in super bra-ket notation as

$$p = \langle\langle E|G_M \cdots G_2 G_1|\rho\rangle\rangle, \tag{6.1}$$

where G_i are super-operators corresponding to the n-qubit gates (circuit layers), $|\rho\rangle\rangle$ is a state preparation, and $\langle\langle E|$ is a POVM effect. In a traditional density-matrix propagation technique, $|\rho\rangle\rangle$ is represented by a real 4^n-dimensional column vector, the G_i by $4^n \times 4^n$ real matrices, and $\langle\langle E|$ by a real 4^n-dimensional row vector. Probability p is then computed by simply multiplying $|\rho\rangle\rangle$ through all of the G_i and taking a final dot product with $\langle\langle E|$. Now consider writing each gate in terms of its error generator. Applying Eq. 3.3 to Eq. 6.1 gives

$$p = \langle\langle E|e^{\Gamma_M} G_M^{(0)} \cdots e^{\Gamma_2} G_2^{(0)} e^{\Gamma_1} G_1^{(0)}|\rho\rangle\rangle. \tag{6.2}$$

If we expand the exponentials as their Taylor series, then

$$\begin{aligned} p &= \langle\langle E| \left(I + \Gamma_M + \frac{\Gamma_M^2}{2!} + \cdots\right) G_M^{(0)} \cdots \\ &\qquad \left(I + \Gamma_2 + \frac{\Gamma_2^2}{2!} + \cdots\right) G_2^{(0)} \\ &\qquad \left(I + \Gamma_1 + \frac{\Gamma_1^2}{2!} + \cdots\right) G_1^{(0)}|\rho\rangle\rangle \qquad (6.3) \\ &= \sum_{k_M \cdots k_1 = 0}^{\infty} \langle\langle E| \frac{\Gamma_M^{k_M}}{k_M!} G_M^{(0)} \cdots \frac{\Gamma_2^{k_2}}{k_2!} G_2^{(0)} \frac{\Gamma_1^{k_1}}{k_1!} G_1^{(0)}|\rho\rangle\rangle. \qquad (6.4) \end{aligned}$$

The sum in the second line expresses p as the sum over all paths from ρ to E, where a path is the product of one term from each G_i's Taylor series, sandwiched between $\langle\langle E|$ and $|\rho\rangle\rangle$. We define the *order of a path* to be the sum of the orders of its constituent Taylor terms, i.e., $\sum_{i=1}^{M} k_i$. We can expand a path into the sum of one or more *elementary paths* by writing each Γ as the sum of "rank-1 Pauli maps" via Eq. 3.4. We define a rank-1 Pauli map to be any mapping $\rho \to \mu P \rho P'$ where P and P' are Paulis and μ is a scalar. Note that the composition of rank-1 Pauli maps is also a rank-1 Pauli map. For later convenience, we rewrite Eq. 3.4 in terms of such maps,

$$\Gamma = \sum_{i=1}^{4^n(4^n-1)} \omega_i R_i, \tag{6.5}$$

where ω_i runs over both the α_j and β_{jk} of Eq 3.4 and R_i is a rank-1 Pauli map - either H_j or one of

the terms of S_{jk} (cf. Eqs. 3.5 and 3.6). We can then expand Γ^k in terms of rank-1 Pauli maps:

$$\Gamma^k = \left(\sum_i R_i\right)^k \tag{6.6}$$

$$= \sum_{i_1+i_2+\cdots i_m=k} \frac{k!}{i_1!i_2!\cdots i_m!} \prod_{q=1}^{m} \omega_{i_q} R_{i_q} \tag{6.7}$$

$$\equiv \sum_{j=1}^{Q(k)} T_j^{(k)}. \tag{6.8}$$

In the second line, we have defined $m = 4^n(4^n-1)$. In the final line, we flatten the multi-index $\{i_l\}_{l=1}^m$ into a single integer index j, and idenfity the rank-1 Pauli maps $T_j^{(k)}$ as the summands in the multinomial expansion of the line above (Eq. 6.7). The number of terms in this expansion is

$$Q(k) = \left(\begin{array}{c} m+k-1 \\ m-1 \end{array} \right). \tag{6.9}$$

We use Eq. 6.8 to replace the $\Gamma_i^{k_i}$ factors in Eq. 6.4 and arrive at an expression for p as the sum of *elementary* paths:

$$p = \sum_{k_M\cdots k_1=0}^{\infty} \sum_{j_M\cdots j_1=1}^{m_{k_M}\cdots m_{k_1}} \langle\langle E| \frac{T_{M,j_M}^{(k_M)}}{k_M!} G_M^{(0)} \cdots \frac{T_{2,j_2}^{(k_2)}}{k_2!} G_2^{(0)} \frac{T_{1,j_1}^{(k_1)}}{k_1!} G_1^{(0)} |\rho\rangle\rangle. \tag{6.10}$$

Each elementary path is indexed by $k_1, k_2, \ldots, k_M$ and by $j_1, j_2, \ldots, j_M$. Let us write these indices compactly as a pair of vector indices, $(\vec{k}, \vec{j})$, whose $2M$ components are the $\{k_i\}$ and $\{j_i\}$.

There are infinitely many paths, and so the manipulations leading to Eq. 6.10 may seem like a fruitless endeavor. However, let us restrict ourselves to the special but common case when the all the $G_i^{(0)}$ are Clifford superoperators, $|\rho\rangle\rangle$ prepares a pure state, and $\langle\langle E|$ projects onto a pure state. Let us now observe:

- The single zero-th order path (composed of all the zero-th order Taylor terms) is the ideal probability - i.e. the result if none of the gates had errors on them. Since these are Clifford superoperators, computing this term is efficient using the stabilizer state framework[3, 4].
- Each elementary path maps $\rho \to C\rho C'$ where C and C' are Clifford operations. This follows from our earlier definition of an elementary path, which made use of the fact that each error generator Γ_i can be written as the sum of rank-1 Pauli maps, and the fact that Pauli operators are isomorphisms on the Clifford group. Together with our assumptions on ρ and E, this observation implies the each elementary path can be efficiently computed using the stabilizer state framework.
- When the errors are small, the α_i and β_{ij} coefficients in the error generators also must be small. Consequently, magnitude of the Taylor terms and elementary paths quickly becomes small with increasing order. This makes the sum of elementary paths amenable to truncation: the number of paths may be reduced to a computable number with a tolerable loss in accuracy.

We find, therefore, that under the stated assumptions the sum in Eq. 6.10 can be truncated in a rigorous way, and each of the retained elementary paths can be computed efficiently. When the magnitudes of the noise are small enough, a number of paths that are tractable to compute yield a sufficiently accurate approximation for p, and it can be used as a proxy for the actual probability. This approximate p is used within the core log-likelihood optimization step of model-based characterization.

Truncation could be performed by simply keeping only the paths up to a given order. This was our first approach, and its relatively simple implementation is its primary advantage. A disadvantage is that it doesn't provide a fixed bound on the errors in the approximated probabilities (the bound depends on the magnitudes of the α_i and β_{ij} coefficients). Said another way, keeping a fixed number of paths obviously doesn't adapt the number of paths to the size of the gate errors. It could be modified to do so, e.g., by computing the maximum possible error in each circuit probability and adjusting the maximum path order accordingly. But choosing a single maximal path order suffers from another more serious problem. We expect that real devices will have some errors that are much larger than others, and this approach treats all errors with the same path order on equal footing. Our path integral approach must be able to deal efficiently with such situations, and keep higher order terms when, and only when, they correspond to large errors. A uniform maximal path order is clearly incapable of such adaptation.

A better approach, and the one that represents the culmination of the LDRD's work, keeps all the paths, regardless of their order, that have a *magnitude* (defined below) greater than a threshold value. This threshold is set on a per-circuit basis, and adjusted so that the approximation error in the outcome probabilities of each circuit falls below a given global threshold η.

An essential part of this approach, then, is to compute an upper bound on the approximation error of each circuit outcome probability. Consider one such probability, p. The path sum in Eq. 6.10 can be decomposed into the sum over the kept paths, that is, the approximate probability p_{approx}, and the sum over the omitted paths, which we denote p_{err}:

$$p = p_{\text{approx}} + p_{\text{err}} \tag{6.11}$$

Let us denote the set of path indices $(\vec{k},\vec{j})$ corresponding to kept paths by $\mathscr{K}$ and the set of omitted path indices by $\mathscr{O}$. Thus, $\mathscr{K}$ is a finite set whereas $\mathscr{O}$ is infinite. The approximation error, p_{err}, is the sum over paths with index $(\vec{k},\vec{j}) \in \mathscr{O}$. Since the entire point of omitting paths is to save the cost of computing them, it is counterproductive to compute p_{err} by summing over omitted paths. We instead compute an upper bound on p_{err} that does not require an explicit omitted-path sum.

We begin by applying the triangle inequality to the definition of p_{err} (the restriction of Eq. 6.10 to paths omitted in the truncation):

$$|p_{\text{err}}| = \left| \sum_{(\vec{k},\vec{j})\in\mathscr{O}} \langle\langle E| \frac{T^{(k_M)}_{M,j_M}}{k_M!} G^{(0)}_M \cdots \frac{T^{(k_1)}_{1,j_1}}{k_1!} G^{(0)}_1 |\rho\rangle\rangle \right| \tag{6.12}$$

$$\leq \sum_{(\vec{k},\vec{j})\in\mathscr{O}} \left| \langle\langle E| \frac{T^{(k_M)}_{M,j_M}}{k_M!} G^{(0)}_M \cdots \frac{T^{(k_1)}_{1,j_1}}{k_1!} G^{(0)}_1 |\rho\rangle\rangle \right| . \tag{6.13}$$

Let $||\cdot||$ denote the vector or matrix 2-norm for preparation and measurement vectors, and gate matrices, respectively. Then, for any superoperator $O = O_2 O_1$, $\langle\langle E|O|\rho\rangle\rangle < ||\langle\langle E|||\cdot||O_2||\cdot||O_1||\cdot|||\rho\rangle\rangle||$ follows from the sub-multiplicative property of the 2-norms. When we apply this to the factor in absolute values within Eq. 6.13, and note that the vector 2-norm of $\langle\langle E|$ and $|\rho\rangle\rangle$ can be at most 1 (their dot product is a probability), and that $||G_i^{(0)}|| = 1$ because these are unitary maps, we're left with

$$|p_{\text{err}}| \leq \sum_{(\vec{k},\vec{j})\in\mathscr{O}} \frac{||T_{M,j_M}^{(k_M)}||}{k_M!} \cdots \frac{||T_{1,j_1}^{(k_1)}||}{k_1!}. \tag{6.14}$$

The right hand side of Eq. 6.14 is a sum over *omitted* paths. This is equal to the same summand, summed over all paths, minus the sum over the kept paths. The sum over all elementary paths,

$$\sum_{(\vec{k},\vec{j})} \frac{||T_{M,j_M}^{(k_M)}||}{k_M!} \cdots \frac{||T_{1,j_1}^{(k_1)}||}{k_1!} \tag{6.15}$$

$$= \sum_{\vec{k}} \left(\sum_{j_M} \frac{||T_{M,j_M}^{(k_M)}||}{k_M!} \cdots \sum_{j_1} \frac{||T_{1,j_1}^{(k_1)}||}{k_1!} \right) \tag{6.16}$$

$$= \sum_{\vec{k}} \left(\frac{\gamma_M^{k_M}}{k_M!} \cdots \frac{\gamma_1^{k_1}}{k_1!} \right) \tag{6.17}$$

$$= \exp \sum_{i=1}^{M} \gamma_i = \prod_{i=1}^{M} \exp(\gamma_i). \tag{6.18}$$

The γ_i factors in Eq. 6.17 are defined by letting γ be the evaluation of Eq. 6.5 with ω_i replaced by its absolute value and R_i is replaced by its norm:

$$\gamma = \sum_{i=1}^{4^n(4^n-1)} |\omega_i| \cdot ||R_i||. \tag{6.19}$$

From Eq. 6.8 it follows that γ^k is not only a bound on $\sum_j^{Q(k)} ||T_j^{(k)}||$ but is *equal* to it, since the sub-multiplicative property applied to $||T_j^{(k)}||$ is saturated for the unitary maps R_i. The derivation of Eq. 6.17 thus follows (with a gate-index subscript added to γ). Equation 6.18 follows immediately from the Taylor series of the exponential function. The final result is that the right hand side of Eq. 6.14, summed over *all* paths, is expressible as a simple function of the absolute values of the various Pauli error coefficients ($|\omega_i|$). These coefficients depend on the model parameters, and so Eq. 6.18 must be recomputed as $\vec{\theta}$ varies. The R_i factors, however, are independent of the model parameters, and so need be computed just once. Also, the $||R_i||$ norm in Eq. 6.19 may be replaced with the matrix 1-norm, since the map is unitary. This is advantageous from an implementation standpoint, as the 1-norm is easier to compute and is used within sparse matrix exponentiation algorithms.

We can use Eq. 6.18 to write the right hand side of Eq. 6.14 as the difference between the sum over all paths and the sum over kept paths in a compact way that can be computed efficiently. We

denote this right hand side by

$$\xi \equiv \prod_{i=1}^{M} \exp(\gamma_i) - \sum_{(\vec{k},\vec{j})\in\mathscr{K}} \frac{||T^{(k_M)}_{M,j_M}||}{k_M!} \cdots \frac{||T^{(k_1)}_{1,j_1}||}{k_1!}, \tag{6.20}$$

and so we can rewrite Eq. 6.14 as

$$|p_{\text{err}}| \leq \xi. \tag{6.21}$$

The error bound ξ is a function of the current parameters, $\vec{\theta}$, in multiple ways. First, the γ and $||T^{(k)}_j||$ factors are explicitly dependent on $\vec{\theta}$. More subtly, the set $\mathscr{K}$ also depends on $\vec{\theta}$, as $\mathscr{K}$ contains all the paths with magnitude greater than a threshold, Ω, and the path magnitudes depend on $||T^{(k)}_j||$ (cf. Eqs. 6.22 and 6.23 below). The key take away is that, $\xi = \xi(\vec{\theta}, \Omega(\vec{\theta}))$ provides a way of tracking how poorly a specific circuit outcome probability is being approximated. We make the circuit-dependence of ξ and Ω explicit using a subscript, e.g. $\xi_C(\vec{\theta}, \Omega_C(\vec{\theta}))$.

We define the *magnitude* of elementary path $(\vec{k}, \vec{j})$ to be

$$\text{PM}(\vec{k}, \vec{j}) = \frac{||T^{(k_M)}_{M,j_M}||}{k_M!} \cdots \frac{||T^{(k_1)}_{1,j_1}||}{k_1!}, \tag{6.22}$$

which is the summand in Eq. 6.20. This real-valued quantity reflects how strong the collective error along the given chain of error terms is and thus how important it is to include in the final path integral. Equation 6.20 is the difference between the sum of *all* possible path magnitudes and the sum of the magnitudes of the paths that are kept. Thus, to make ξ small using as few paths as possible, the best strategy is to keep the paths with the largest magnitudes. For each circuit C, we choose a *path-magnitude threshold* Ω_C and keep all paths with magnitude greater than or equal to this threshold. Thus, the set of kept-paths, $\mathscr{K}$, for each outcome of C is given by

$$\mathscr{K}_C = \{(\vec{k}, \vec{j}) \,\text{s.t.}\, \text{PM}(\vec{k}, \vec{j}) \geq \Omega_C\}. \tag{6.23}$$

We choose Ω_C small enough that $\xi(\Omega_C) < \eta$ for each circuit C (where we have simplified notation by dropping the $\vec{\theta}$ dependence). In this way, we compute as few paths as possible in order to attain a given guarantee on the approximation error. The values of Ω_C (one value per circuit) define a specific path truncation.

The approach given above finds a sufficient set of paths for computing circuit outcome probabilities within a prescribed error tolerance η. It does this at a single point $\vec{\theta}$ in model parameter-space, i.e., for a concrete instantiation of all the gate errors, since the path magnitudes depend on $\vec{\theta}$. The question then emerges: How do we optimize the log-likelihood *and* control the approximation error in the outcome probabilities? We attempted several approaches to this before settling on the one we currently use.

One approach is to recompute the per-circuit thresholds Ω_C, and thereby the kept paths $\mathscr{K}_C$, at each objective function evaluation. This approach suffered from at least two issues: 1) it is expensive to compute the Ω_C, and 2) the objective function is then only approximately consistent from point to point. The first issue is a practical one, as performing the optimization in this way may negate all the performance gains of using the path integral technique in the first place. The

second issue makes the LM optimization less robust - the objective function is more like a stochastic function and may require the specialized methods of stochastic optimization.

A second approach is to stop the optimization whenever a step is attempted that doesn't meet the desired error tolerances. We found this approach to be problematic because the optimization would not proceed very far before attempting such a step. This is not surprising, since only the *minimal* number of paths needed to achieve the given η are kept, so any substantial step in parameter space is likely to void the ability of the selected paths to achieve η.

We have settled on an approach that sets two different error tolerances η and η'. The first, η, is used when constructing the path sets $\mathscr{K}_C$, as before. The second, η', is more relaxed ($\eta' > \eta$), and is used as a stopping criterion during the optimization. When the optimizer would have *accepted* a step that fails to achieve η' the LM optimization is halted, new path sets are obtained (using η), and the optimization is resumed. This continues until the optimization converges within the region where the probabilities are approximated within the tolerance η'. (We restrict the number of allowed restarts to 5 for practical purposes.)

We have tested this approach throughout its development on simulated data. The tests proceed similarly to those using the density-matrix-propagating circuit simulation method - the only difference is in how the circuit outcome probabilities are computed and the additional stopping criterion and restarts mentioned above. We test the method using cloud-noise models restricted to Hamiltonian and Pauli-stochastic errors with $w_I = 2$, $w_g = 0$, and $r_g = 1$. Data is simulated for 2-10 qubits with finite sample error using $N = 10,000$ samples per circuit. We only consider shallow circuits ($L = 1$ in all but the 1-qubit case) as this is the regime we expect the technique to perform the best. The data generating model is chosen exactly as it was for the validation in section 5.2. We choose $\eta = 0.01 = 1/\sqrt{N}$ and $\eta' = 10\eta = 0.1$. A comparison of the best-fit objective function values (in units of N_σ) using the approximated probabilities as well as those using the exact (density matrix propagated) probabilities, is given in Fig. 6-1.

We find overall that the path-integral method works well in the low-error shallow-circuit regime studied. It consistently finds a best-fit solution with low N_σ that, when the comparison is possible, is close to the result of the exact probability simulation. Figure 6-2 shows that the number of CPU hours required is drastically reduced from that of the full density matrix forward simulator, reducing run times by approximately two orders of magnitude. At the very least, this technique holds promise as one that quickly finds a good seed for other methods.

We believe there is still room for exploration and improvement in this area. In particular, η and η' are the absolute errors allowed in the *probability* values rather than in the objective function. It seems potentially more useful to set error tolerances in terms of the objective function, as this is what is ultimately being optimized. Secondly, the optimization method is still somewhat ad-hoc, and we do not believe we have found what is clearly the best way to perform it. Further algorithmic development and testing will prove beneficial. Finally, we expect the bound ξ of Eq. 6.20 is loose, and can be improved upon.

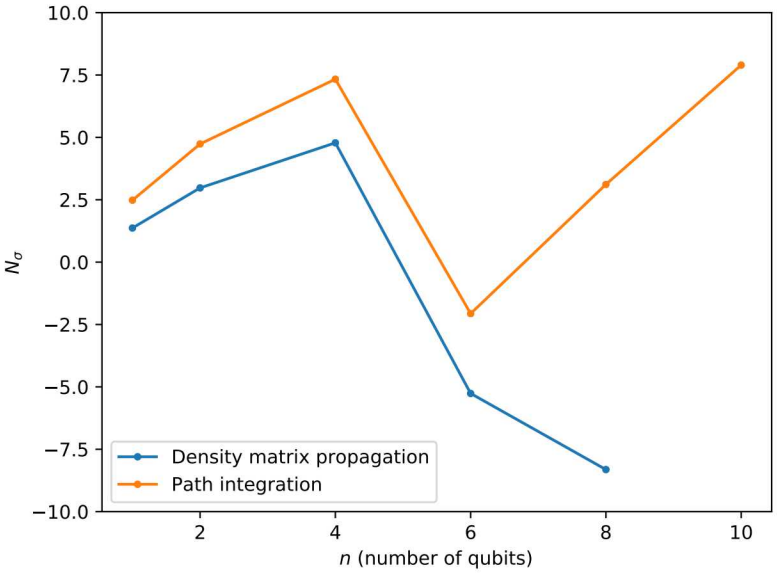

Figure 6-1. Comparison of our tomography protocol using approximate (path-integral) and standard (density matrix propagation) forms of circuit simulation. N_σ values are computed by applying Wilks' theorem to the true and estimated models (as in Fig. 5-1), and quantify our confidence in the validity of the estimated model. The estimated model *should* be valid, given the source of the data is a cloud-noise model of the same type as that which generated the data. The maximum L value used for the 1-qubit results was 16; for all other qubit numbers only $L = 1$ circuits were used. The low N_σ values in all cases confirm that using both circuit simulation methods the tomography protocol was able to reconstruct the data generated by the true model well. The path-integral approach performs slightly less well, but tracks closely with the exact circuit simulation method up to $n = 8$ qubits, and uses many fewer resources (allowing it to be extended to $n = 10$).

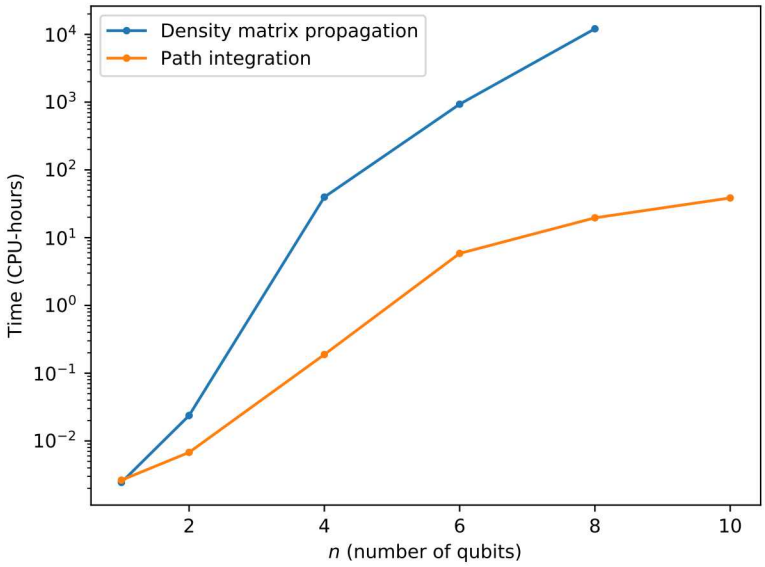

Figure 6-2. Comparison of the time needed to perform each of the tests of Fig. 6-1. Times are given in CPU-hours, and we find that the path-integral approach to circuit simulation requires roughly two orders of magnitude less time than the standard approach of density-matrix propagation.

7. SUMMARY AND OUTLOOK

This LDRD was successful in achieving it's primary goal of developing an tomographic protocol that can be applied to many-qubit processors. We have described that method in this report. We have shown that our method efficiently represents the noise in a many-qubit QIP using so-called cloud-noise models. The circuits needed to probe the parameters of a cloud-noise model can be found efficiently, using a newly developed, scalable approach to circuit selection. Optimization algorithms that have been tuned and customized are able to reliably optimize over the space of model parameters, finding a good best-fit model when one exists. Our testing on existing QIPs demonstrates that cloud-noise models are able to capture a substantial amount of the noise in existing hardware, and offers promising prospects for future hardware. Finally, a novel method for computing circuit outcome probabilities was developed that ultimately allows our tomographic method to be scaled to 10-15 qubits. This technique leverages certain common contexts, such as the ideal gates being Clifford operations, to overcome the exponential-in-qubit-count behavior endemic to traditional methods.

Overall, this 3-year research effort has been a great success. The project ends having reached its goal of creating a new efficient and scalable form of QIP tomography. When the project began the state of the art was limited to performing tomography on at most 2 qubits. Pushing this figure to 10-15 qubits is an accomplishment that was seen as an incredibly lofty goal three years ago.

While the project largely followed the track set out at its beginning, there were several unexpected lessons we learned along the way.

- Optimizing circuit simulators for tomography is difficult and time-consuming work. We underestimated the amount of time and effort required to tweak and tune what in many respects is a standard and well-studied problem: the classical simulation of a quantum circuit. The problem, however, changes dramatically when there is not one but thousands of circuits that need to be simulated repeatedly - sometimes tens of thousands of times. To tackle this problem, different time vs. memory tradeoffs, especially those surrounding the caching of intermediate values, become critical. We ended up only being able to simulate 6-8 qubits using standard circuit-simulation techniques rather than 8-10 as we initially expected, and only after putting substantial effort into solving the problem.

- There are many nuances to scaling up actual QIPs. At the beginning of this project we expected to have access to many QIPs with 4-10 qubits, as many research groups seemed on the brink of scaling up their well-oiled 2-qubit platforms. This was not the case. There were to us a great many underappreciated complexities to making even marginally functional many-qubit QIPs in practice, many related to material science and device fabrication. In the end, we had fewer available QIPs to test, and the larger devices were significantly more noisy than we expected (cf. section 5.3). We learned that noise in actual devices may be

much more varied than we though, and attempted to mitigate the paucity of available test systems by increasing the flexibility of our models and leaning more heavily on simulations.

The follow-on funding opportunities for this work are numerous, and some have already been secured. In future work, we plan to focus on more extensive experimental verification (more 4-10 qubit devices are becoming available now) and on further honing the path-integral approach detailed in section 6. Experimental systems are very diverse, and a closer study of more experimental systems will allow us to further tailor our flexible cloud-noise models to push current hardware to be better in its next generation. The path-integral approach could be the start of something what could grow into breakthrough algorithms needed to scale characterization methods to still more qubits. Thanks to the work done in this LDRD, we have all the necessary groundwork to pursue these two exciting avenues and continue to make important contributions to the field of quantum computing.

REFERENCES

[1] Robin Blume-Kohout. A taxonomy of small errors (in preparation).

[2] Robin Blume-Kohout, John King Gamble, Erik Nielsen, Kenneth Rudinger, Jonathan Mizrahi, Kevin Fortier, and Peter Maunz. Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography. *Nat. Commun.*, 8:14485, 2017.

[3] Jeroen Dehaene and Bart De Moor. Clifford group, stabilizer states, and linear and quadratic operations over GF(2). *Phys. Rev. A*, 68(4):042318, 2003.

[4] Héctor J. García and Igor L. Markov. Simulation of quantum circuits via stabilizer frames, 2017.

[5] Thomas Häner and Damian S. Steiger. 0.5 petabyte simulation of a 45-qubit quantum circuit. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2017.

[6] Erik Nielsen, John King Gamble, Kenneth Rudinger, Travis Scholten, Kevin Young, and Robin Blume-Kohout. Gate set tomography, 2020.

[7] Erik Nielsen, Kenneth Rudinger, Timothy Proctor, Antonio Russo, Kevin Young, and Robin Blume-Kohout. Probing quantum processor performance with pyGSTi. *Quantum Science and Technology*, 5(4):044002, jul 2020.

[8] MA Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[9] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[10] Timothy Proctor, Melissa Revelle, Erik Nielsen, Kenneth Rudinger, Daniel Lobser, Peter Maunz, Robin Blume-Kohout, and Kevin Young. Detecting, tracking, and eliminating drift in quantum information processors. *arXiv preprint arXiv:1907.13608*, 2019.

[11] Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout. Measuring the capabilities of quantum computers, 2020.

[12] Mark K. Transtrum and James P. Sethna. Improvements to the levenberg-marquardt algorithm for nonlinear least-squares minimization, 2012.

DISTRIBUTION

Hardcopy—External

Number of Copies	Name(s)	Company Name and Company Mailing Address

Hardcopy—Internal

Number of Copies	Name	Org.	Mailstop
1	D. Chavez, LDRD Office	1911	0359

Email—Internal (encrypt for OUO)

Name	Org.	Sandia Email Address
Technical Library	01177	libref@sandia.gov

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.