Sample-efficient verification of continuously-parameterized quantum gates for small quantum processors

Ryan Shaffer[1,3], Hang Ren[1,3], Emiliia Dyrenkova[2,3], Christopher G. Yale[4], Daniel S. Lobser[4], Ashlyn D. Burch[4], Matthew N. H. Chow[4,5,6], Melissa C. Revelle[4], Susan M. Clark[4], and Hartmut Häffner[1,3]

[1]Department of Physics, University of California, Berkeley, CA 94720, USA
[2]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, USA
[3]Challenge Institute for Quantum Computation, University of California, Berkeley, CA 94720, USA
[4]Sandia National Laboratories, Albuquerque, NM 87123, USA
[5]Department of Physics and Astronomy, University of New Mexico, Albuquerque, NM 87131, USA
[6]Center for Quantum Information and Control, University of New Mexico, Albuquerque, NM 87131, USA

Most near-term quantum information processing devices will not be capable of implementing quantum error correction and the associated logical quantum gate set. Instead, quantum circuits will be implemented directly using the physical native gate set of the device. These native gates often have a parameterization (e.g., rotation angles) which provide the ability to perform a continuous range of operations. Verification of the correct operation of these gates across the allowable range of parameters is important for gaining confidence in the reliability of these devices. In this work, we demonstrate a procedure for sample-efficient verification of continuously-parameterized quantum gates for small quantum processors of up to approximately 10 qubits. This procedure involves generating random sequences of randomly-parameterized layers of gates chosen from the native gate set of the device, and then stochastically compiling an approximate inverse to this sequence such that executing the full sequence on the device should leave the system near its initial state. We show that fidelity estimates made via this technique have a lower variance than fidelity estimates made via cross-entropy benchmarking. This provides an experimentally-relevant advantage in sample efficiency when estimating the fidelity loss to some desired precision. We describe the experimental realization of this technique using continuously-parameterized quantum gate sets on a trapped-ion quantum processor from Sandia QSCOUT and a superconducting quantum processor from IBM Q, and we demonstrate the sample efficiency advantage of this technique both numerically and experimentally.

Ryan Shaffer: ryan.shaffer@berkeley.edu, Current affiliation: AWS Quantum Technologies, Seattle, WA 98170, USA. Work done prior to joining Amazon.

1 Introduction

Verifying the correct operation of quantum computations is an essential step toward building a reliable and scalable quantum information processing device [1]. Most commonly, quantum computations are broken down into fundamental building blocks known as *quantum gates*, which may include gates such as the well-known Hadamard, Pauli, and CNOT operations. Verifying the behavior of a device's physically-realizable gates, known as a *native gate set*, is a primary area of research in this field. The most complete techniques for gate verification belong to a family of techniques known as *tomography*. Such techniques include quantum state tomography [2, 3], quantum process tomography [4], and gate set tomography [5, 6]. Tomographic techniques produce a complete characterization of a quantum operation, which provides a detailed mathematical description of the errors present in the system. However, tomography is extremely resource-intensive, and

although techniques exist to improve its scalability somewhat [7, 8], its cost still typically scales exponentially with qubit count.

In contrast, *benchmarking* techniques for verifying quantum gates are typically resource-efficient and in principle can be scaled to much larger qubit counts than tomographic techniques. These techniques notably include randomized benchmarking (RB) [9, 10] and more scalable variants such as cycle benchmarking [11], direct RB [12], and mirror RB [13], which involve executing randomized circuits which are equivalent to the identity. Additional techniques include as cross-entropy benchmarking (XEB) [14] and matchgate benchmarking [15, 16], which compare the ideal and experimental output probabilities of random quantum circuits. Benchmarking provides an incomplete characterization of a quantum system, typically producing a small number of values which attempt to characterize the average error rate of particular operations performed by the system. But as the name implies, such techniques are particularly useful when attempting to compare the performance of distinct devices, since they provide metrics which are ostensibly hardware-agnostic. For example, benchmarking techniques may provide an estimate of the average error rate of executing a CNOT gate or of a device's average state preparation and measurement (SPAM) error.

Because many quantum algorithms and especially quantum error correction schemes are expressed in terms of particular fixed sets of gates – most commonly, the Clifford+T family, which is universal for quantum computation – much of the benchmarking literature is focused on verifying the operation of these fixed one-qubit and two-qubit gates, or their device-native equivalents. However, near-term quantum devices are unlikely to implement large-scale quantum error correction [17], and instead will implement circuits directly using the physical native gate set of the device. These native gates are often not limited to the fixed gate set used by error correction schemes, but rather have a parameterization which provides the ability to perform a continuous range of operations. For example, a single-qubit operation may frequently be parameterized as $R(\theta, \varphi)$, where θ is the rotation angle and φ is the axis of rotation. Multi-qubit operations may also be parameterized. The typical multi-qubit gate for trapped-ion devices, based on the Mølmer-Sørensen interaction [18, 19], can be parameterized as $MS(\theta, \varphi)$, where θ can be interpreted as the effective rotation angle in the multi-qubit space, and φ is the effective multi-qubit axis of rotation [20]. In many instances, compiling quantum circuits using continuously-parameterized native gate sets can produce more efficient compilations on physical devices than when limited to fixed gate sets [21].

Systematic and efficient verification techniques for continuously-parameterized quantum gates, therefore, are a key ingredient for near-term quantum computers to reliably take advantage of the full scope of physically-realizable operations. Standard RB-based techniques are sample-efficient and often scalable to large system sizes, but these protocols typically have restrictions on the gate sets (e.g., limited to only Clifford gates) and therefore are not generally applicable to this task. A notable exception is a recent proposed variant of mirror RB for universal gate sets [22], which provides a scalable RB-like protocol for gate sets which may include non-Cliffords (although some restrictions on the gate set remain) and was demonstrated on a 27-qubit device. Previous work has also developed an interleaved RB technique to estimate the fidelity of an arbitrary gate [23, 24], which allows for verification of a particular instance of a parameterized gate, but not across the range of allowed parameters. The XEB protocol, which was used to demonstrate quantum computational supremacy on a 53-qubit device [25], uses random quantum circuits formed from an arbitrary continuously-parameterized gate set with randomly-chosen parameters. However, because XEB circuits have a broad probability distribution over measurement outcomes, XEB is less sample-efficient than RB, which ideally concentrates all of the probability on a single measurement outcome.

In this work, we discuss the application of the *randomized analog verification* (RAV) technique [26] to the task of verifying an arbitrary gate set consisting of continuously-parameterized quantum gates. By concentrating most of the measurement probability on a single outcome (like RB), RAV provides a sample-efficient protocol for verification of gate sets across the range of allowed parameters. As we show in this work, RAV can be practically implemented for quan-

tum processors of up to approximately 10 qubits. In Section 2, we provide an overview of the RAV technique and compare it to XEB; we describe the stochastic compilation scheme used in constructing the RAV sequences; and we provide details on the experimental setup of the trapped-ion quantum processor, the Quantum Scientific Computing Open User Testbed (QSCOUT) operated by Sandia National Laboratories [27], including the technical details of the functionality required to support the continuously-parameterized gate set and large circuit depths used in the RAV sequences. In Section 3, we provide numerical simulations demonstrating the conditions under which we expect RAV to provide a sample efficiency advantage over XEB, and we report experimental demonstrations of this efficiency advantage on the QSCOUT trapped-ion device and on a superconducting quantum processor from the publicly-available IBM Q service [28]. We conclude with additional discussion of these results in Section 4.

2 Methods

2.1 Randomized analog verification for continuously-parameterized quantum gates

2.1.1 RAV protocol

The verification technique introduced in this work is a gate-based adaptation of the *randomized analog verification* (RAV) protocol for analog quantum simulators [26]. When applied to analog quantum simulations, the RAV protocol consists of running randomized analog sequences of subsets of terms of a target Hamiltonian. In particular, a set of unitary operators is chosen consisting of short, discrete time steps of each of the terms of the target Hamiltonian. A randomly-generated sequence of these operations is then applied, which evolves the system to some arbitrary state. Next, an approximate inverse of this sequence, generated using the same set of unitary operators by using a stochastic compilation protocol (see Section 2.2), is applied to the system, which returns it to the initial state with high probability.

Because current gate-based, non-error-corrected quantum computers are realized by carefully tuning the underlying analog interactions to implement quantum gates with the highest fidelity possible, it is natural to adapt the RAV protocol for use in verifying the behavior of gate-based devices with continuously-parameterized native gates. The RAV protocol is described in Figure 1. The RAV protocol, like XEB, constructs random sequences of *layers*, each of which consists of some fixed number of each of the device's native gates in some randomly-chosen order. Random parameter values are then provided to each of these gates, which allows the protocol to verify the behavior of the device across the range of allowable parameter values for each gate. But unlike XEB, which proceeds by sampling directly from the output of these random sequences, RAV appends a compiled sequence of layers which approximately inverts the initial sequence. Sequences of varying lengths are generated and run on the target device. Finally, an average *error per layer* is extracted from the results.

Schematics of the XEB and RAV protocols are displayed in Figure 2, which illustrate the fact that the primary difference of RAV from XEB is the inversion sequence which returns the system nearly to the initial state.

2.1.2 Fidelity estimates using XEB and RAV

Given a single XEB sequence on an n-qubit system, we can approximate the resulting fidelity as

$$\hat{F}_{\mathrm{XEB}} = \frac{\sum_x P(x)Q(x) - \frac{1}{N}}{\sum_x P(x)^2 - \frac{1}{N}} \tag{1}$$

where we have simplified the linear cross-entropy fidelity formula [14] for the case of a single circuit.[1] Here, $P(x)$ represents the classically-computed ideal output probability distribution for the sequence, $Q(x)$ is the observed sample probability of obtaining measurement result x, and $N = 2^n$ is the dimension of the system. $\hat{F}_{\mathrm{XEB}}$ is constructed such that its observed value for a single circuit might not fall within the range $[0, 1]$. But in general, the expected fidelity of the ideal output state (i.e., if $P(x) = Q(x)\ \forall\ x$) is 1, and the expected fidelity of a maximally-mixed output state is 0.

[1] The fidelity estimates for XEB and RAV discussed in this section are strictly valid only when averaging over ensembles of circuits; however, we write our expressions in terms of only a single circuit in order to make the analysis more readable.

1. Choose a gate set G. Typically this will be the native gate set of the device. Each gate in G is specified as a parameterized unitary (with zero or more parameters), along with the set of allowed target qubits.

2. Choose a layer design L_G as follows. For each gate in G, specify the number of such gates that will appear in each layer, along with allowed ranges for each parameter. A layer is generated from L_G by the following steps:

   (a) Select each gate the specified number of times.

   (b) Choose parameters for each gate uniformly at random from the allowed range.

   (c) Choose target qubit(s) for each gate uniformly at random from the allowed set.

   (d) Randomly permute the order of the selected gates.

3. Generate the RAV sequences as follows. Choose a range of initial layer counts for the RAV sequences $M_0 = (m_{0,\min}, \ldots, m_{0,\max})$ which can be expected to cover a reasonable range of fidelity loss. For example, $m_{0,\min}$ should be small enough to have expected sequence fidelity near 1, and $m_{0,\max}$ should be large enough to have expected sequence fidelity near the fully-decayed limit (but not too large). Then, for each $m_0 \in M_0$:

   (a) Generate a sequence of m_0 random layers, each generated independently according to the layer design L_G.

   (b) Calculate the product of this sequence U. Generate an approximate compilation using STOQ (see Section 2.2), where the target unitary is $U^\dagger$ and the instruction set consists of layers generated by L_G. The resulting inversion sequence has length m_{inv} and error $\epsilon = 1 - \frac{1}{N^2}|\text{Tr}(VU)|^2$, where V is the product of the generated inversion sequence and $N = 2^n$ is the dimension of the n-qubit system. If necessary, repeat the STOQ compilation until a desired ϵ is achieved.

   (c) Concatenate the initial random sequence and the inversion sequence to produce the final RAV sequence with layer count $m = m_0 + m_{\text{inv}}$. This sequence ideally leaves the system in the initial state x_0 with probability $P(x_0) = 1 - \epsilon$ when averaged over all initial states.

4. Run K shots of each RAV sequence, with randomly chosen initial states, and record the probability $Q(x_0)$ of measuring the system to be in the initial state after running the sequence. Calculate $\hat{F}_{\text{RAV}}$ for each sequence as specified in Equation 2.

5. Plot $\hat{F}_{\text{RAV}}$ as a function of the total layer count m. Assuming negligible state preparation errors, fit the results to an appropriate fidelity loss curve based on the properties of the experimental error, e.g., an exponential decay curve $\hat{F}_{\text{RAV}} = \alpha^m$ or a Gaussian curve $\hat{F}_{\text{RAV}} = \alpha^{m^2}$ (see Section 2.1.4 for further discussion).

Figure 1: Description of the randomized analog verification (RAV) protocol for continuously-parameterized gate sets.

To derive a formula for the fidelity of a single RAV sequence on an n-qubit system, we start with the XEB formula in Equation 1. We then note that, by construction of the RAV sequence, $P(x_0) = 1 - \epsilon$, where x_0 is the initial state (and expected final state) of the RAV sequence and ϵ is the approximation error of the inversion sequence. After some simplification (see Appendix A), we arrive at the following expression for the approximate RAV sequence fidelity:

$$\hat{F}_{\text{RAV}} = \frac{Q(x_0) - \frac{1}{N}}{P(x_0) - \frac{1}{N}}. \tag{2}$$

We use the hat on the symbols $\hat{F}_{\text{XEB}}$ and $\hat{F}_{\text{RAV}}$ to emphasize that they are only estimates of fidelity based on a single circuit instance. To obtain reliable information about the fidelity, these results must be aggregated over many circuit instances. In addition, the use of the linear cross-entropy to estimate $\hat{F}_{\text{XEB}}$ as in Equation 1 is only strictly true in the limit of large circuit depth and when averaged over ensembles of random circuits [14]; therefore, the expression for $\hat{F}_{\text{RAV}}$ in Equation 2 is subject to the same limitation. However, as we demonstrate by numerical simulations (see Figure 4), the derived variance of $\hat{F}_{\text{RAV}}$ agrees well with observed data even in the regime of small qubit count $2 \leq n \leq 8$ and moderate circuit depth of $10 \leq m \leq 30$ layers.

The quantity estimated by $\hat{F}_{\text{XEB}}$ and $\hat{F}_{\text{RAV}}$ is known as the *depolarization fidelity* [25]. For a circuit whose ideal pure output state is $|\psi\rangle$ and whose execution is subject to purely depolarizing errors, the true mixed output state can be defined in terms of a depolarization parameter λ as

$$\rho_\lambda = (1 - \lambda)|\psi\rangle\langle\psi| + \frac{\lambda}{N} I \tag{3}$$

where $\lambda \in [0, 1]$ is the fraction to which the output state is depolarized. The depolarization fidelity of this state is then defined as $F = 1 - \lambda$. This quantity is related to, but distinct from, the typical state fidelity $f = \left(\text{Tr}\sqrt{\sqrt{\rho_\lambda}|\psi\rangle\langle\psi|\sqrt{\rho_\lambda}}\right)^2$. As derived in [25], the average depolarization fidelity $\overline{F}$ is related to the average state fidelity $\overline{f}$ as $\overline{f} = \overline{F} + (1 - \overline{F})/N$. For a fully-depolarized system, the average depolarization fidelity $\overline{F} = 0$, whereas the average state fidelity $\overline{f} = \frac{1}{N}$. We also note that because depolarization fidelity is linearly related to state fidelity, the shape of the loss curve of either quantity will maintain the same character (e.g., exponential or Gaussian).

2.1.3 Sample efficiency of RAV

Intuitively, we expect that measuring the success of RAV sequences should be more sample-efficient than measuring the success of XEB sequences. This is because RAV requires estimating the output probability $Q(x_0)$ of only the initial state (under the assumptions used to derive Equation 2), whereas XEB requires sampling from the full output probability distribution $Q(x)$. Additionally, in the case where the error is small, we expect the final state of a RAV sequence to be close to a basis state, which minimizes the quantum projection noise associated with the measurement.

More concretely, we can demonstrate this sample efficiency advantage by calculating the statistical variance associated with measurement of $\hat{F}_{\text{RAV}}$ and $\hat{F}_{\text{XEB}}$ for a single sequence. By making several simplifying assumptions (see Appendix B for full details), we can approximate the variance of these quantities as

$$\begin{aligned}\text{Var}[\hat{F}_{\text{RAV}}] \approx \frac{1}{K}\left(\frac{1}{(1-\epsilon)-\frac{1}{N}}\right)^2 \times \\ \left[(1-\lambda)(1-\epsilon) + \frac{\lambda}{N}\right] \times \\ \left[1 - (1-\lambda)(1-\epsilon) - \frac{\lambda}{N}\right]\end{aligned} \tag{4}$$

$$\begin{aligned}\text{Var}[\hat{F}_{\text{XEB}}] \approx \frac{1}{K}\left(\frac{1}{\frac{1}{2}-\frac{1}{N}}\right)^2 \left[\frac{1}{2}\left(\frac{\lambda}{N}\right)\left(1 - \frac{\lambda}{N}\right) + \right. \\ \left. \frac{1}{3}(1-\lambda)\left(1 - \frac{2\lambda}{N}\right) - \frac{1}{4}(1-\lambda)^2\right]\end{aligned} \tag{5}$$

where K is the number of independent experimental shots taken for the given sequence and $\epsilon \ll 1$ is the error in the compiled inversion of the RAV sequence.

We plot the standard deviation (i.e., $\sqrt{\text{Var}[\hat{F}_{\text{RAV}}]}$ and $\sqrt{\text{Var}[\hat{F}_{\text{XEB}}]}$) of these fidelity estimates in Figure 3 for $2 \leq n \leq 16$ qubits and $0 \leq \lambda \leq 1$, assuming $K = 100$ and $\epsilon = 0.04$. The mean of the RAV and XEB fidelity estimates are in agreement in all cases, but we observe from these plots that RAV fidelity estimates have a lower standard deviation than XEB in all cases, with the advantage tending to shrink as λ and n increase. This implies that fewer RAV repetitions are necessary in order to get an equivalently-precise estimate of the fidelity of a given sequence.

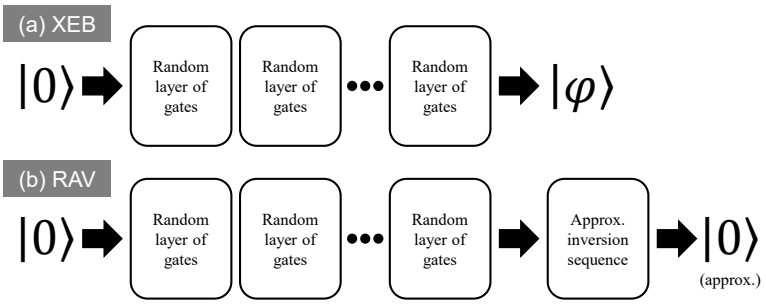


Figure 2: Schematics of sequences used in (a) cross-entropy benchmarking (XEB) and (b) randomized analog verification (RAV) protocols for continuously-parameterized quantum gates. The states denoted as $|0\rangle$ can in general be any computational basis state. In XEB, the state denoted as $|\varphi\rangle$ is the ideal result of applying the XEB sequence to the initial state, and is in general far from any computational basis state. Each protocol includes a sequence of randomly-generated layers of native gates with randomly-chosen parameters. In RAV, this is followed by a compiled sequence of layers which is an approximate inverse of the initial sequence and returns the system to the initial state with high probability.

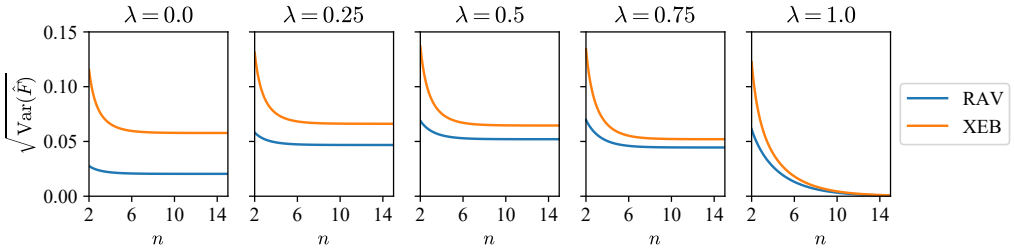


Figure 3: Ideal standard deviation of $\hat{F}_{\mathrm{RAV}}$ and $\hat{F}_{\mathrm{XEB}}$ measurements for n-qubit systems as calculated analytically by Equation 4 and Equation 5, where $\lambda = 0$ corresponds to an ideal output state, $\lambda = 1$ corresponds to a maximally-mixed output state (see Equation 3), and we assume a RAV inverse approximation error of $\epsilon = 0.04$. All plots use $K = 100$ independent measurement shots per sequence.

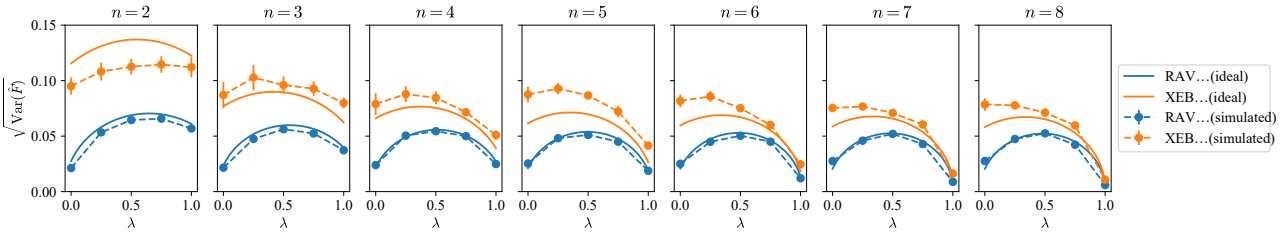


Figure 4: Comparison of the precision of fidelity measurements using RAV and XEB at various levels of fidelity loss, where $\lambda = 0$ corresponds to an ideal output state and $\lambda = 1$ corresponds to a maximally-mixed output state (see Equation 3). The "simulated" data points, marked by circles, represent the observed standard deviation of 100 simulated measurements of $\hat{F}_{\mathrm{RAV}}$ (or $\hat{F}_{\mathrm{XEB}}$) for a set of representative RAV (or XEB) sequences on an n-qubit system with $10 \leq m \leq 30$ layers per circuit. Error bars indicate standard error of the mean across the set of sequences; for the RAV data points, the error bars are smaller than the data markers. The "ideal" solid curves represent the ideal standard deviation of $\hat{F}_{\mathrm{RAV}}$ and $\hat{F}_{\mathrm{XEB}}$ measurements for the given n and λ as calculated analytically by Equation 4 and Equation 5, assuming a RAV inverse approximation error of $\epsilon = 0.04$. All plots use $K = 100$ independent measurement shots per sequence.

We note that the RAV precision advantage is largest in the regime where λ is small, since this is exactly the regime where the final state of a RAV sequence is near a basis state, which minimizes the quantum projection noise associated with the computational basis measurement.

As a further illustration, we perform a simulation of RAV and XEB fidelity measurements across various regimes of fidelity decay assuming a purely depolarizing channel. We generate representative RAV and XEB sequences for systems with $2 \leq n \leq 8$ qubits, and we calculate the standard deviation of fidelity measurements on these sequences for $0 \leq \lambda \leq 1$. These plots, shown in Figure 4, show good agreement between the calculated standard deviation for the RAV circuits and the standard deviation extracted from simulations. We observe some quantitative disagreement for the XEB circuits, which is likely because the assumptions behind Equation 5 are not necessarily valid in the $2 \leq n \leq 8$ and $10 \leq m \leq 30$ parameter regime used in these simulations (see Appendix B). Nonetheless, we observe qualitative agreement between the calculated and simulated results for the XEB circuits as well as for the RAV circuits, and we observe that the variance of the RAV fidelity estimates is lower than the variance of the XEB fidelity estimates in all cases. This supports the existence of the RAV sample efficiency advantage, even in the absence of the simplifying assumptions used to derive Equation 4 and Equation 5.

2.1.4 Fitting RAV fidelity loss curves

The RAV protocol description (see Figure 1) does not prescribe a particular fit function, since the shape of the fidelity loss depends on properties of the generated circuits and of the experimental error sources. Random quantum circuits of large-enough depth (i.e., deep enough that the circuits form approximate 2-designs [29]) effectively transform coherent errors into global depolarizing noise [14, 30], which would result in a pure single exponential decay. This argument applies to both XEB circuits and RAV circuits, which both consist of sequences of random layers. However, for sequences which are not of sufficient depth, the fidelity loss curve contains additional terms and cannot be accurately modeled as a single exponential decay. Attempting to do so can easily result in an overestimate of per-layer

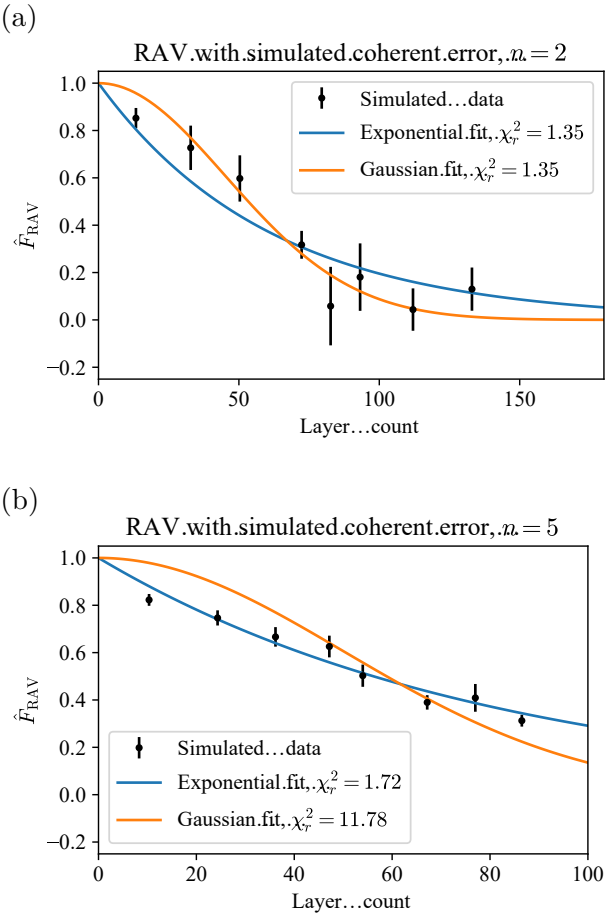


Figure 5: Simulated results of RAV sequences under coherent error showing different fidelity loss behavior. Coherent error is implemented in these simulations as a fixed over-rotation of 0.15 radians ($\approx \pi/20$) applied to each physical single-qubit and two-qubit gate. Each data point represents the mean result of six distinct circuits. Error bars represent the standard error of the mean. Fits are a single-parameter exponential curve $\hat{F}_{\mathrm{RAV}} = \alpha^m$ and a single-parameter Gaussian curve $\hat{F}_{\mathrm{RAV}} = \alpha^{m^2}$. Goodness of fit is reported as the reduced chi-squared statistic χ_r^2. (a) Simulated results of 50 RAV sequences on $n = 2$ qubits. Sequences are the same as used for experimental demonstration in Section 3.2. (b) Simulated results of 50 RAV sequences on $n = 5$ qubits. Sequences are the same as used for numerical demonstration in Section 3.1.

fidelities [31, 32]. In addition, we note that any real device is subject to a variety of coherent and stochastic error sources which on their own would not act as a depolarizing channel. So it is not likely that experimental results under various realistic noise sources using relatively small-depth sequences will show good agreement with the idealized results under the assumption of purely depolarizing noise, or that the fidelity estimates extracted under such an assumption will be accurate.

To demonstrate this concretely, we show in Figure 5 the results of simulating RAV sequences under a fixed coherent error, which we model as a fixed over-rotation of 0.15 radians ($\approx \pi/20$) applied to each physical single-qubit and two-qubit gate. Sequences are generated using the native gate set described in Section 3.1. In Figure 5(a), we observe that $\hat{F}_{\mathrm{RAV}}$ for these two-qubit RAV sequences appears to follow something in between a Gaussian curve and an exponential curve. In Figure 5(b), for this set of five-qubit RAV sequences, we observe that $\hat{F}_{\mathrm{RAV}}$ appears to be fit more closely by an exponential than a Gaussian, but still deviates notably from the ideal exponential decay. These results indicate that the sequences do not satisfy the conditions described above in order to transform the coherent errors into global depolarizing noise. We further note that in an experimental setting, it is likely not feasible to know the shape of the fidelity loss curve in advance, since the properties of the noise are not fully known and may change with time.

For our experimental results in Section 3.2, we use different fits for the fidelity loss depending on the observed shape of the data. Because this decision is partially dependent on the properties of the particular device, RAV (as the name implies) should be considered a *verification* protocol rather than a *benchmarking* protocol. It provides quantitative metrics about the correctness of a device's operation, but one cannot interpret these metrics in a device-independent way without further assumptions about the types of errors that exist in the device.

2.2 Stochastic compilation using continuously-parameterized quantum gates

We now discuss our technique for compiling the approximate inversion sequence as part of RAV sequence generation. Compiling an inversion sequence for a given random quantum circuit in a non-trivial manner (i.e., other than simply reversing and inverting the original random sequence) into a sequence of continuously-parameterized gates is a difficult problem that in general is infeasible to solve exactly. To produce the inversion portion of the RAV sequences, we introduce a stochastic protocol for approximate quantum unitary compilation, which we abbreviate as STOQ. We note that this protocol generalizes a similar technique used for variational quantum compilation algorithms [33, 34].

The process of compilation requires specification of the unitary operation to be compiled, known as the *target unitary*. This is the 2^n-dimensional unitary operator U implementing some desired effect on the n-qubit system. In the case of RAV, the target unitary is the inverse of the product of the initial randomly-generated sequence of layers.

The set of gates used for the compilation may, in general, be fixed or parameterized. *Fixed gates*, such as Cliffords, are discrete operations that can be represented as a fixed unitary matrix. In contrast, *parameterized gates*, such as rotations, are continuous operations that can be represented as a unitary matrix with one or more continuously-variable parameters. The allowed set of gates for the compilation may then consist of some combination of fixed and parameterized gates. For an n-qubit system, the *instruction set* (often called *native gate set*) is a set of fixed gates and/or parameterized gates that represent the fundamental set of operations that can be physically applied to the system.

For a protocol such as RAV, it is desirable that gates occur in the sequence in specific patterns, which we refer to as *layers*. In this case, we can define the instruction set in terms of these layers of fixed and/or parameterized native gates, such that the resulting compilation will be a sequence of layers. Each layer consists of a fixed number of each type of native gate, where each gate is assigned a randomly-chosen parameter value (within some allowed parameter range) and the order of the gates within the layer is also randomly chosen. In the following description of the STOQ algorithm, we refer to the components of an instruction set as *instructions*, where each instruction can be either a single gate or a layer, depending on the definition of the instruction set

for the given problem.

Given a target unitary U and an instruction set G, then, the goal of an *approximate compilation* protocol is to find a sequence of instructions $\{G_1, \ldots, G_M\}$ such that the product $G_M G_{M-1} \cdots G_1$ is as close as possible to U for some reasonable choice of distance metric. Note that this definition does not require any particular closeness of approximation, but it does require that the quality of approximation can be measured. That is, given some appropriate distance metric which defines a distance d between the sequence product $G_M G_{M-1} \cdots G_1$ and the target unitary U, an approximate compilation procedure treats d as the value of a cost function to be minimized.

The STOQ protocol for approximate compilation proceeds according to the pseudocode displayed in Figure 6. Intuitively, the STOQ algorithm can be thought of as a randomized exploration of the full set of possible n-qubit unitary operators (or the subset that can be generated by the instruction set G, if G is not universal), using a technique known as Markov chain Monte Carlo (MCMC) search [35]. The algorithm is always initialized with an empty sequence, meaning that it always starts from the identity operator in the search space. At each iteration, a random step is proposed, in which an item is either added to or removed from the sequence. If this step brings the product of the sequence closer to the target unitary as determined by the cost function, it is accepted; otherwise, it is either accepted or rejected with some probability, where the probability of accepting such "bad" steps decreases with each iteration. The algorithm continues until some maximum number of iterations is reached, at which point the final cost can be evaluated and the sequence either kept or discarded, depending on the accuracy requirements of the given problem.

One critical component of the algorithm is the choice of an appropriate and efficiently-computable cost function. Naturally, the cost function should be a distance measure between the the target unitary U and the unitary V which is the product of the currently-compiled sequence. One commonly-used and operationally-relevant choice, used also in variational quantum compilation approaches [33, 34], is a distance metric defined using the norm of the Hilbert-Schmidt inner product,

$$D_{\mathrm{HS}}(U, V) = \left|\mathrm{Tr}(V^\dagger U)\right|, \tag{6}$$

which is related to the fidelity of a process [36]. We therefore use the cost function

$$\texttt{Cost}(U, V) = 1 - \frac{1}{2^n} D_{\mathrm{HS}}(U, V), \tag{7}$$

noting that $\texttt{Cost}(U, V)$ ranges from 0 to 1 and vanishes if and only if U and V are equivalent up to a global phase.

Additional technical details and discussion of STOQ are provided in Appendix C, including demonstrations of using STOQ for compilation of sequences to approximately implement time-evolution unitaries, as well as for approximate compilation of randomly-generated unitaries.

```
function StochasticCompilation
(params U, G, num_iterations):
  sequence := []
  beta := 0
  cost := Cost(U, Prod(sequence))
  for i in 1 to num_iterations:
    beta := IncreaseBeta(beta)
    new_sequence := RandomChange(sequence, G)
    new_cost := Cost(U, Prod(new_sequence))
    if Accept(cost, new_cost, beta):
      sequence := new_sequence
      cost := new_cost
  return seq
```

Figure 6: Pseudocode for STOQ stochastic compilation algorithm. The inputs to the algorithm are the target unitary `U`, the parameterized instruction set `G`, and the number of iterations to perform `num_iterations`. The algorithm is described in Section 2.2, with additional implementation details provided in Appendix C.2.

2.3 QSCOUT experimental setup

The Quantum Scientific Computing Open User Testbed (QSCOUT) is a quantum processor based on trapped ions housed at Sandia National Laboratories [27]. For the experiments shown here, two $^{171}\mathrm{Yb}^+$ ions were used, in which the qubit states are defined by the hyperfine 'clock' transition of a $^{171}\mathrm{Yb}^+$ ion, $^2\mathrm{S}_{1/2}$ $|\mathrm{F}{=}0, m_F = 0\rangle$ ($|0\rangle$) and $|\mathrm{F}{=}1, m_F = 0\rangle$ ($|1\rangle$). The ions are controlled via Raman transitions using a pulsed 355 nm laser in the counter-propagating configuration [37, 38], where one arm is a "global" beam that spans all qubits and the other arm consists of up to 32 individual addressing beams, each of which illuminates a single ion [39]. All

beams, individual or global, are controlled via the Sandia developed "Octet" coherent control hardware with complete two-tone frequency, phase, and amplitude control via rf pulses. The individual addressing beams are created by a specialized multichannel acousto-optic modulator (AOM) from L3Harris, which divides a single laser beam into 32 beams and propagates each through a separate AOM crystal.

The single- and two-qubit gates used in the system are all generated via Raman transitions and are parameterized. The single-qubit gates utilize the appropriate individual beam with two tones applied the AOM, generating the necessary transitions in what is known as a co-propagating configuration. Gates about an equatorial axis on the Bloch sphere are physical gates called $R(\theta, \varphi)$, and defined by both θ, which is determined by the duration of the pulse, and φ, determined by the relative phase of the two tones in the Raman transition. The pulse amplitudes are square-shaped and gapless, meaning there is no "off" time between single qubit gates. Fidelities for physical single-qubit gates, $R(\pi/2, 0)$ and $R(\pi/2, \pi/2)$ have been estimated to be $99.5 \pm 0.3\%$ using a variety of techniques including gate set tomography. The gates used for this work do not contain any form of compensation, such as SK1 [40], simplifying the bare gate error analysis and reducing the data acquisition time to limit the effects of drift, especially in the case of significant layer sizes. $R_Z(\theta)$ gates are applied virtually by Octet and seen by the qubits as a cumulative phase shift.

Two-qubit gates in the system are Mølmer-Sørensen (MS) interactions of the form $XX(\theta) = e^{-i\frac{\theta}{2}\sigma_X \otimes \sigma_X}$ and are also parameterized. They are defined by a desired phase, φ and angle, θ. The MS gate pulses have a Gaussian-shaped amplitude and are composed of one tone on the global beam and two tones each on the individual beams, generating Raman transitions which are detuned symmetrically from a red and blue motional sideband pair.

Unlike the single-qubit gates, the MS gate has a fixed duration of 200 μs, and the rotation angle θ is determined by the global beam amplitude, accounting for distortions and saturation effects in the global beam amplifier and AOM. In addition, negative rotation angles are generated by changing the relative phase on one of the two ions by π radians. For the purposes of this demonstration, all $MS(\theta, \varphi)$ gates had $|\theta| \leq \pi/10$, and for these small values of θ, calibrations suggest deviations in the resultant rotation angle of $\lesssim 8\%$.

Additionally, the MS gates also account for the AC Stark effect through the use of frame rotations, which are virtual Z rotations, applied during the MS gate to cancel phase accumulation from the AC Stark effect. As the global beam and individual beams both contribute to phase shifts caused by the AC Stark shift, the frame rotation also changes depending on the desired θ. These are calibrated to within $\pm 3.5 \times 10^{-3}$ radians, or ± 0.2 degrees, for the range of θ used.

The MS gates are performed in a counter-propagating beam configuration while the single-qubit gates are performed in a co-propagating configuration. Because the relative phase stability between the counter-propagating beams is less stable than that of co-propagating beams, intermixing gates from the two configurations leads to unpredictable phase relationships. To combat this instability, we perform basis transformations on all two-qubit gates. We first surround the MS gate with counter-propagating single-qubit $\pi/2$ gates to transform an XX interaction into a ZZ interaction [41]. We then further surround those with co-propagating single-qubit gates to bring the interaction back onto an equatorial axis. The desired phase of the MS gate, φ, is then introduced through the respective phase of these co-propagating single-qubit gates. When including the basis transformation gates, we estimate fidelities for $MS(\pi/2, 0)$ to be $97 \pm 1\%$.

Due to the nature of the RAV sequences, some extra consideration was needed to deal with non-standard sequences. Low-level pulse data is compressed and stored in a series of lookup tables (LUTs) in Octet's programmable logic for fast readout of data-intensive gate sequences.[2] The topology of these LUTs and the compression scheme is designed to leverage redundant gate information common to a wide array of quantum circuits. Because of the numerous unique gate calls in RAV sequences, the compression ratio is limited and more LUT storage is required. While storage was increased for certain LUTs using a custom addressing scheme for dense packing of data (not limited to byte-write boundaries), fi-

[2]Gate sizes are comparatively larger than other control systems because of the designed flexibility for multi-parameter spline-based modulation [27].

nite memory availability is still a limitation.

Increasing the LUT storage was supplemented by a compilation technique developed to partially reprogram large segments of the LUTs mid circuit. In this work, the RAV sequences used numerous virtual R_Z gates. Due to their virtual nature, R_Z gates are typically on the order of 10 ns, and continuous streaming of raw data for such short gates exceeds the maximum data throughput supported by the device. The compiler was set up to run recursively, essentially breaking long circuits into smaller pieces based on where the LUT capacity was exceeded. To prevent underflow conditions, partial reprogramming data was placed after gates with long durations by strategic adjustment of the initial boundaries determined by the compiler.

3 Results

3.1 Numerical demonstrations

To demonstrate the sample efficiency advantage of RAV, we generated 50 RAV and 50 XEB sequences of varying lengths for a five-qubit system. We generated these sequences using a continuously-parameterized native gate set $\{R(\theta,\varphi), R_Z(\theta), MS(\theta,\varphi)\}$, where these operations are defined in matrix form using the computational basis as follows:

$$R(\theta,\varphi) = \begin{bmatrix} \cos\frac{\theta}{2} & -ie^{i\varphi}\sin\frac{\theta}{2} \\ ie^{-i\varphi}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{8}$$

$$R_Z(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \tag{9}$$

$$MS(\theta,\varphi) = \tag{10}$$
$$\begin{bmatrix} \cos\frac{\theta}{2} & 0 & 0 & -ie^{-i2\varphi}\sin\frac{\theta}{2} \\ 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} & 0 \\ 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} & 0 \\ -ie^{i2\varphi}\sin\frac{\theta}{2} & 0 & 0 & \cos\frac{\theta}{2} \end{bmatrix}$$

We chose this gate set because it aligns most directly with the native gate set of the QSCOUT trapped-ion processor.[3] Specifically, $R(\theta,\varphi)$ and

[3] In our implementation of RAV and XEB sequence generation, we used the Python representation of these parameterized operations provided by Sandia at https://gitlab.com/jaqal/qscout-gatemodels/.

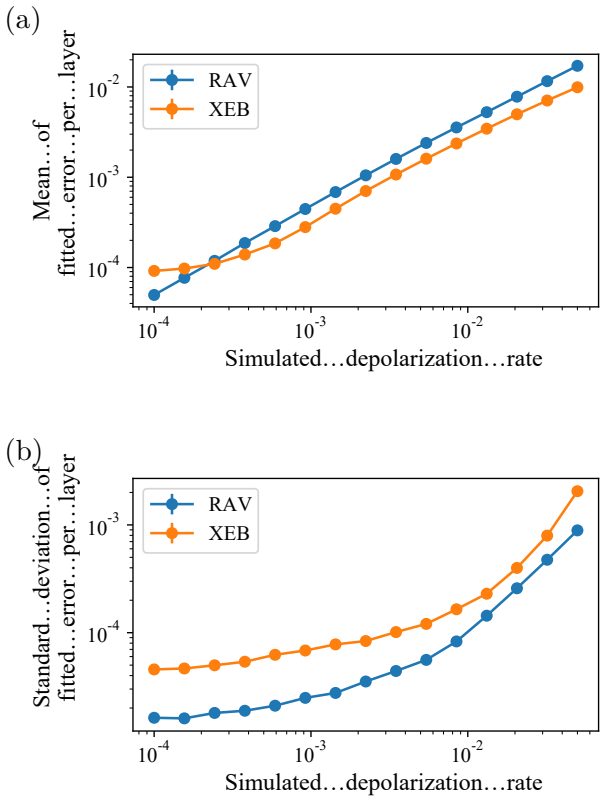


Figure 7: Statistics of fitted error rates from simulations of five-qubit RAV and XEB runs, using an exponential fit $\hat{F} = \alpha^m$ to extract the result from each run, and using the quantity $1-\alpha$ as the error rate. Each data point represents the fitted error rate statistics of 100 independent simulations, using 50 RAV or XEB circuits of up to 800 layers with $K = 100$ shots per circuit, as described in Section 3.1. The sets of RAV and XEB sequences are of matching lengths; i.e., for each of the 50 RAV sequences, an XEB sequence was generated with the same number of layers. Error bars, which are smaller than the data markers, represent the standard error of the mean across 10 independent repetitions. Simulated depolarization rate is implemented as the amount of depolarization per $\theta = \pi/2$ rotation via single-qubit $R(\theta,\varphi)$ gate or $\theta = \pi/20$ rotation via two-qubit $MS(\theta,\varphi)$ gate, where the total amount of depolarization is proportional to the rotation angle θ. Simulated $R_Z(\theta)$ gates are unaffected by depolarization rate. (a) Mean fitted error per layer as a function of simulated depolarization rate. (b) Standard deviation of fitted error per layer as a function of simulated depolarization rate.

$MS(\theta, \varphi)$ are implemented by the QSCOUT device as native one-qubit and two-qubit physical operations, and $R_Z(\theta)$ is implemented by the QSCOUT device as a virtual one-qubit operation which requires no physical interaction with the qubits.

We generated the sets of sequences such that the total layer counts of the RAV and XEB sequences are equivalent. Because RAV sequence lengths are unpredictable due to the stochastic inversion compilation process, we first generated 50 RAV sequences over a range of sequence lengths up to 800 layers. For each RAV sequence, we then generated an XEB sequence of exactly the same length.

Each generated layer consists of three $R^{(i)}(\theta, \varphi)$ gates, three $R_Z^{(i)}(\theta)$ gates, and one $MS^{(i,j)}(\theta, \varphi)$ gate. The target qubit(s) for each gate, represented by the superscript indices, are chosen uniformly at random from the set of all qubits in the system. A layer is constructed by first choosing random parameter values for each of these seven gates. Values for each θ rotation angle are chosen uniformly at random in the interval $[-\pi/10, \pi/10]$,[4] and values for each φ axis angle are chosen uniformly at random in the interval $[-\pi, \pi]$. After the parameter values are chosen, the order of the gates within the layer is then randomly permuted, which produces the layer that is ultimately used in the sequence.

Figure 7 depicts the mean and standard deviation of error rates per layer obtained via both RAV and XEB for sequences under varying simulated depolarization rates, using an exponential fit $\hat{F} = \alpha^m$ and using the quantity $1 - \alpha$ as the error rate. We observe that, over a wide range of depolarization rates, the mean fitted error rates from RAV and XEB are closely aligned, but the error rates estimated by RAV have a significantly smaller standard deviation than those obtained via XEB, indicating that RAV provides more precise information about the overall error rate of these sequences. We also observe that this advantage is more significant in the regime of lower depolarization rates. For a depolarization rate around 10^{-2}, the RAV error rate estimates are roughly twice as precise as the XEB error rate estimates, whereas around 10^{-4}, they are roughly three times as precise.

We note that because the RAV sample efficiency advantage comes partially from reduced quantum projection noise due to measurement, we expect that we will find the largest advantage when operating in the early part of the RAV decay curve (which in this simulation is realized by lower depolarization rate). This is the regime where the RAV sequence results remain closest to a basis state, where quantum projection noise is minimized. However, as described previously, RAV continues to have a sample efficiency advantage in the regime of larger error rates because it requires estimating the output probability of only a single state, rather than sampling from the full output probability distribution as required by XEB.

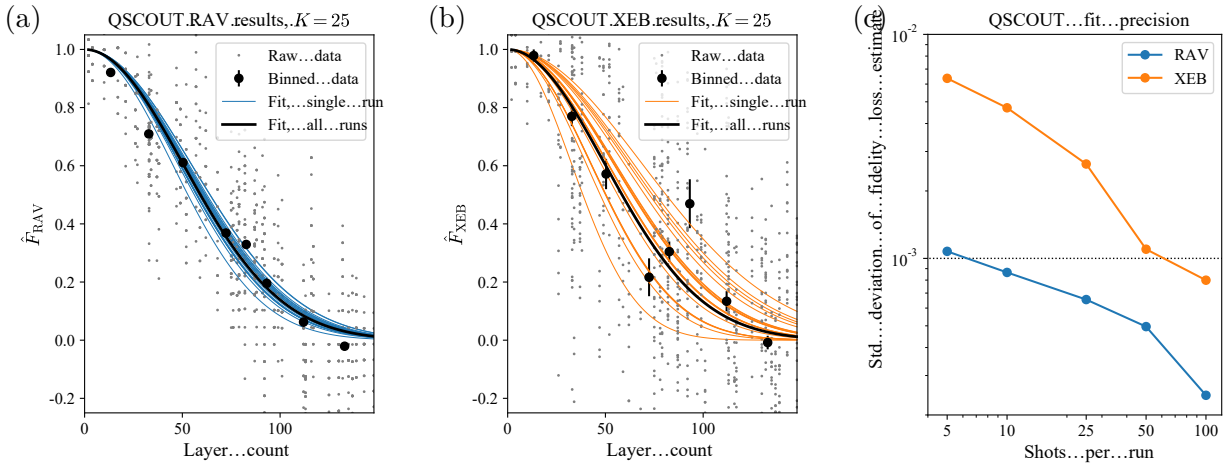
[4] Allowed parameter ranges are chosen with consideration to the performance of the inverse compilation via STOQ. The stochastic search process resembles a stochastic gradient descent, meaning that each generated layer should ideally result in a "small" change to the circuit. Empirically, for this gate set, we find that this tends to be satisfied when each rotation angle $|\theta| \ll \pi$. Here we chose the allowed parameter range $|\theta| \leq \pi/10$ because this resulted in reasonably good convergence behavior.

3.2 Experimental demonstrations

To demonstrate the RAV sample efficiency advantage experimentally, we generated RAV and XEB sequences of varying lengths for a 2-qubit system using the same native gate set as in Section 3.1. In this subsection, we report the results from executing these sequences on quantum processors from QSCOUT and IBM Q.

3.2.1 QSCOUT trapped-ion processor

As a first experimental demonstration, we executed 50 RAV and 50 XEB sequences on the two-qubit trapped-ion quantum processor at the Quantum Scientific Computing Open User Testbed (QSCOUT) operated by Sandia National Laboratories [27]. Details of the experiment are provided in Section 2.3. We note that this device directly implements the parameterized native gate set $\{R(\theta, \varphi), R_Z(\theta), MS(\theta, \varphi)\}$ that we used to generate these sequences.



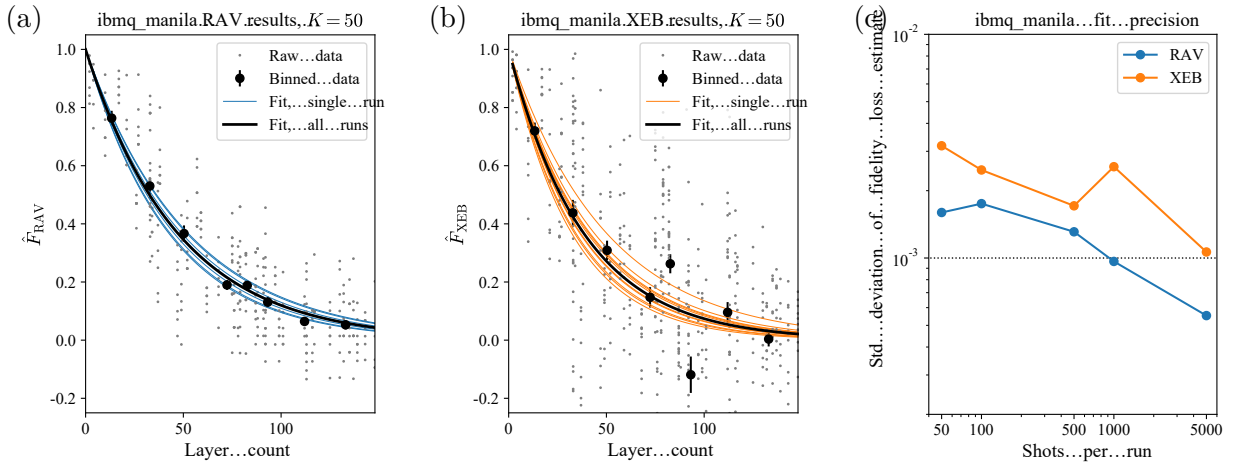
(d)

Shots per run	*RAV: fidelity loss estimate* Mean ($\times 10^{-2}$)	SD ($\times 10^{-2}$)	$\frac{\text{SD}}{\text{Mean}}$	*XEB: fidelity loss estimate* Mean ($\times 10^{-2}$)	SD ($\times 10^{-2}$)	$\frac{\text{SD}}{\text{Mean}}$	$\frac{\text{XEB } \frac{\text{SD}}{\text{Mean}}}{\text{RAV } \frac{\text{SD}}{\text{Mean}}}$
$K = 5$	1.407	0.108	0.076	1.506	0.637	0.423	5.53
$K = 10$	1.404	0.087	0.062	1.463	0.470	0.321	5.21
$K = 25$	1.403	0.065	0.047	1.399	0.264	0.189	4.04
$K = 50$	1.402	0.050	0.035	1.353	0.110	0.081	2.30
$K = 100$	1.402	0.024	0.017	1.345	0.080	0.059	3.40

Figure 8: Experimental results from QSCOUT for two-qubit RAV and XEB runs. Each run consists of 50 sequences executed K times each. (a,b) Results of 20 independent RAV (or XEB) runs using $K = 25$ shots per sequence. Raw data points indicate the fidelity estimate $\hat{F}_{\mathrm{RAV}}$ (or $\hat{F}_{\mathrm{XEB}}$) for K shots of a single sequence, calculated according to Equation 2 (or Equation 1). Each binned data point (with error bars) represents the mean (and standard error of the mean) of $\hat{F}_{\mathrm{RAV}}$ or $\hat{F}_{\mathrm{XEB}}$ for a set of six sequences across all 20 runs. Thin curves are single-parameter Gaussian fits $\hat{F} = \alpha^{m^2}$ for the data points from each run, which is chosen based on empirical goodness of fit (see Section 3.2.1 for details). The thick curve is the mean of the individual fit curves. (c) Standard deviation of the fidelity loss estimate $\sqrt{1-\alpha}$ for RAV and XEB runs for various values of K. Smaller standard deviation indicates a more precise estimate of the fidelity loss estimate. (d) Fidelity loss estimate statistics for RAV and XEB runs for various values of K.

Figure 8(a) and Figure 8(b) show the results of 20 independent runs of the entire set of RAV and XEB sequences, using $K = 25$ shots per experiment. The same set of sequences was used for each run, but we performed the 20 independent runs in order to be able to calculate the mean and standard deviation of the estimated fidelity loss obtained from fitting each run to a Gaussian curve $\hat{F} = \alpha^{m^2}$. (For experimental simplicity, 500 executions were performed for each sequence, and we then separated the shot-level results and interpreted them as $500/K$ independent runs of K shots each.) Based on empirical goodness of fit (see Section 2.1.4), we choose to fit the data to an Gaussian curve $\hat{F} = \alpha^{m^2}$ (RAV $\chi_r^2 = 17.4$, XEB $\chi_r^2 = 3.64$) rather than a exponential curve $\hat{F} = \alpha^m$ (RAV $\chi_r^2 = 64.3$, $\chi_r^2 = 19.4$).

It is clear visually in Figure 8(a) and Figure 8(b) that the XEB fit curves vary significantly more from the mean than the RAV fit curves. The standard deviations of the fidelity loss estimates for various values of K are plotted in Figure 8(c), and the corresponding statistics are tabulated in Figure 8(d). As expected, we observe that the the fidelity loss estimates obtained via RAV runs have a significantly smaller relative standard deviation (by a factor of 2.30 to 5.53) than those obtained from XEB runs. Since the standard deviation of the fidelity estimate goes as $1/\sqrt{K}$ (which is supported by the data in Figure 8(c)), this implies that XEB would require approximately 5 to 30 times as many experimental shots as RAV to produce a fidelity loss estimate for this device with equivalent precision. For example, Figure 8(c) illustrates that the RAV $K = 5$ runs provide a more precise fidelity loss estimate than the XEB $K = 50$ runs.



(d)

Shots per run	*RAV: fidelity loss estimate* Mean ($\times 10^{-2}$)	SD ($\times 10^{-2}$)	$\frac{\text{SD}}{\text{Mean}}$	*XEB: fidelity loss estimate* Mean ($\times 10^{-2}$)	SD ($\times 10^{-2}$)	$\frac{\text{SD}}{\text{Mean}}$	$\frac{\text{XEB } \frac{\text{SD}}{\text{Mean}}}{\text{RAV } \frac{\text{SD}}{\text{Mean}}}$
$K = 50$	2.101	0.160	0.076	2.572	0.318	0.123	1.62
$K = 100$	2.514	0.175	0.070	2.444	0.248	0.101	1.45
$K = 500$	2.182	0.131	0.060	2.563	0.171	0.067	1.11
$K = 1000$	2.194	0.097	0.044	2.927	0.256	0.087	1.99
$K = 5000$	2.047	0.055	0.027	2.434	0.106	0.044	1.62

Figure 9: Experimental results from IBM Q `ibmq_manila` device for two-qubit RAV and XEB runs. Each run consists of 50 sequences executed K times each. (a,b) Results of 10 independent RAV (or XEB) runs using $K = 50$ shots per sequence. Raw data points indicate the fidelity estimate $\hat{F}_{\mathrm{RAV}}$ (or $\hat{F}_{\mathrm{XEB}}$) for K shots of a single sequence, calculated according to Equation 2 (or Equation 1). Each binned data point (with error bars) represents the mean (and standard error of the mean) of $\hat{F}_{\mathrm{RAV}}$ or $\hat{F}_{\mathrm{XEB}}$ for a set of six sequences across all 10 runs. Thin curves are single-parameter exponential decay fits $\hat{F} = \alpha^m$ for the data points from each individual run, which is chosen based on empirical goodness of fit (see Section 3.2.2 for details). The thick curve is the mean of the individual fit curves. (c) Standard deviation of the fidelity loss estimate $1 - \alpha$ for RAV and XEB runs for various values of K. Smaller standard deviation indicates a more precise estimate of the fidelity loss. (d) Fidelity loss estimate statistics for RAV and XEB runs for various values of K.

3.2.2 IBM Q superconducting processor

To provide further experimental results, we executed the same 50 RAV and 50 XEB sequences on the publicly-available `ibmq_manila` superconducting processor from IBM [28]. This device does not directly implement the parameterized native gate set $\{R(\theta,\varphi), R_Z(\theta), MS(\theta,\varphi)\}$. To adapt the sequences for this device, we translated the sequences from the original parameterized native gate set into the instruction set that is accepted by the IBM Q framework. More specifically, $R(\theta,\varphi) = U(\varphi, \theta - \pi/2, \pi/2 - \theta)$ and $MS(\theta,\varphi) = R_Z(-\varphi)^{\otimes 2} R_{XX}(\theta) R_Z(\varphi)^{\otimes 2}$, where $R_{XX}(\theta)$ is itself a composite gate that can be implemented through multiple native gates. $R_Z(\theta)$ is implementable directly and does not need to be translated.

Figure 9(a) and Figure 9(b) show the results of 10 independent runs of the entire set of RAV and XEB sequences, using $K = 50$ shots per experiment. The shapes of the XEB fit curves can be seen to have a larger spread than the RAV fit curves, which again illustrates the smaller uncertainty in estimating fidelity via RAV vs. XEB. Based on empirical goodness of fit (see Section 2.1.4), we choose to fit the data to an exponential curve $\hat{F} = \alpha^m$ (RAV $\chi_r^2 = 1.46$, XEB $\chi_r^2 = 4.76$) rather than a Gaussian curve $\hat{F} = \alpha^{m^2}$ (RAV $\chi_r^2 = 24.9$, XEB $\chi_r^2 = 17.8$). The experiment is repeated for varying numbers of shots per sequence, as is shown in Figure 9(c). Figure 9(d) tabulates the specific statistics.[5] For some of the

[5] A total of eight RAV runs and five XEB runs on the IBM Q device are excluded from the statistics reported in Figure 9 due to abnormally elevated error rates during these particular runs. The qualitative results of the analysis are not affected.

experiments, the relative standard deviation of the fidelity loss estimates from the XEB experiments is up to twice that of the RAV experiments, as expected.

Based on the gate error rates published by IBM at the time the experiments were run, we can calculate that the error per layer of our RAV and XEB sequences should be about 1.57×10^{-2}, which indicates that our experiments overestimated the true error rate. We also note that the IBM Q results in Figure 9(c) do not demonstrate the expected scaling of the standard deviation of the fidelity estimate as $1/\sqrt{K}$. We believe that this is because the true error rate of the `ibmq_manila` device was fluctuating during the course of the runs, which were spread over the course of several hours. This led directly to increased variance in fitted fidelity loss estimates for some of the experiments; for example, comparing to the QSCOUT results in Figure 8(c), we clearly see much higher variance in the IBM Q results despite using the same set of RAV and XEB sequences. Therefore, the variances of our fidelity loss estimates are including the effects of these physical fluctuations in addition to the inherent variance from the RAV and XEB estimates. We expect that runs in which all of the sequences are executed in rapid succession would help to reduce the effect of such fluctuations.

4 Discussion

We note that because of its advantages in sample efficiency, RAV may be particularly useful in the context of frequent calibration runs for devices with continuously-parameterized gates. Minimizing the number of experimental shots per calibration run reduces the downtime of a device due to calibration, and therefore increases its availability for more useful work. Although RAV itself is not a calibration scheme, it can be used to rapidly obtain an estimate of average error over the device's continuously-parameterized gate set (more rapidly than techniques based on XEB, as shown in this work), which could in turn be used as feedback to a calibration technique or as verification of a completed calibration. In this context, the sample efficiency of RAV provides a notable advantage over XEB. We observed from the two-qubit QSCOUT experimental data (see Figure 8) that achieving a fidelity estimate precision of 0.125% required only $K = 10$ shots of each of the 50 RAV sequences vs. $K = 100$ shots of each of the 50 XEB sequences. Each shot took an average of 23 ms experimentally. The corresponding total runtime for a single RAV run is 11.5 s (for 500 total shots), whereas for a single XEB run the total runtime is 115 s (for 5000 total shots). Typical operation of the two-qubit QSCOUT device involves a calibration run approximately every 20 minutes, and so reducing the post-calibration verification step from 115 s to 11.5 s would result in a 10% increase in the available computational time of the device. We also note that future systems will have larger numbers of qubits and smaller error rates. We expect that both of these aspects will increase the time needed for verification, such that the practical advantage of RAV over XEB in terms of runtime could be substantial.

In this work, we have demonstrated RAV and XEB experimentally on continuously-parameterized gate sets of both a trapped-ion system from Sandia QSCOUT and a superconducting system from IBM Q. In particular, the gate sets used in this demonstration more closely align with the native physical gates of trapped-ion systems. In QSCOUT, one circuit layer contains a single physical two-qubit gate, where the parameterized θ is directly associated with physical inputs to that gate; however, on the IBM Q system, one circuit layer contains two physical two-qubit gates, and the parameterized θ is instead passed into the surrounding single-qubit gates. As such, the particular RAV construction demonstrated here provides an estimate of the fidelity loss that may provide more direct assessment of physical two-qubit controls for the trapped-ion QSCOUT system, yet still provides an estimate of the fidelity loss for the superconducting IBM Q system that can be compared to their published error rate. Additionally, RAV could easily be adapted to include gates more closely aligned with the native physical gates of any system, including superconducting systems. We also detail the experimental realization of continuously-parameterized two-qubit gates (and relevant calibration bounds to these gates) on the QSCOUT system. We note that additional control-system developments, in the form of extra storage and partial reprogramming, were needed to support the lengthy sequences of unique gate instantiations inherent to verifica-

tion techniques for continuously-parameterized gate sets. With these capabilities, the RAV and XEB sequences were reasonably practical to experimentally implement on a trapped-ion system such as QSCOUT.

We have demonstrated the generation of RAV sequences on systems of up to $n = 8$ qubits. The bottleneck in the RAV sequence generation is the compilation of the approximate inversion sequence via STOQ, which scales poorly with system size (see Appendix C.5) and is unlikely to be feasible for $n \gg 10$ qubits. For example, the inversion compilation for each $n = 8$ RAV sequence used for Figure 4 took approximately an hour to generate using a laptop computer. However, this is not an inherent limitation of RAV. If a more efficient technique can be applied to generating the approximate inversion sequence, then RAV sequences could be generated for larger systems. We believe that one promising area of future work would be to adapt the efficient inversion techniques from mirror RB [13] to the context of RAV and verification of continuously-parameterized gates. For efficient mirroring, this would likely require some restrictions on the construction of layers of the generated RAV circuits, such as requiring that each layer consists only of Clifford gates in order to be efficiently classically simulable.

Code and Data Availability

An open-source implementation of RAV sequence generation and the STOQ compilation protocol [42] is freely available at `https://github.com/rmshaffer/stoq-compiler`. Data and sequences used to generate the plots in this paper are available upon request from the corresponding author.

Acknowledgements

R.S., H.R., E.D., and H.H. acknowledge support from the Challenge Institute for Quantum Computation (CIQC) via the NSF Quantum Leap Challenge Institute (QLCI) program under grant number OMA-2016245, from the Army Research Office under grant number W911NF-18-1-0170, and from the NSF STAQ project under grant number PHY-1818914. R.S. acknowledges support from the QISE-NET fellowship under NSF award DMR-1747426, as well as from the National Defense Science and Engineering Graduate (NDSEG) fellowship under contract FA9550-11-C-0028 and awarded by the Department of Defense, Air Force Office of Scientific Research, 32 CFR 168a.

This material was also funded in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research Quantum Testbed Program. 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. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan). SAND2023-03170J.

We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

References

[1] Alexandru Gheorghiu, Theodoros Kapourniotis, and Elham Kashefi. "Verification of Quantum Computation: An Overview of Existing Approaches". Theory of Computing Systems **63**, 715–808 (2019).

[2] P. Walther, K. J. Resch, T. Rudolph, E. Schenck, H. Weinfurter, V. Vedral, M. Aspelmeyer, and A. Zeilinger. "Experimental one-way quantum computing". Nature **434**, 169–176 (2005).

[3] Marcus Cramer, Martin B. Plenio, Steven T. Flammia, Rolando Somma, David Gross, Stephen D. Bartlett, Olivier Landon-Cardinal, David Poulin, and Yi Kai Liu. "Efficient quantum state tomography". Nature Communications **1**, 149 (2010).

[4] Seth T. Merkel, Jay M. Gambetta, John A. Smolin, Stefano Poletto, Antonio D. Córcoles, Blake R. Johnson, Colm A. Ryan, and Matthias Steffen. "Self-consistent quantum process tomography". Physical Review A **87**, 062119 (2013).

[5] 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". Nature Communications **8**, 14485 (2017).

[6] Erik Nielsen, John King Gamble, Kenneth Rudinger, Travis Scholten, Kevin Young, and Robin Blume-Kohout. "Gate Set Tomography". Quantum **5**, 557 (2021).

[7] A. Shabani, R. L. Kosut, M. Mohseni, H. Rabitz, M. A. Broome, M. P. Almeida, A. Fedrizzi, and A. G. White. "Efficient measurement of quantum dynamics via compressive sensing". Physical Review Letters **106**, 100401 (2011).

[8] B. P. Lanyon, C. Maier, M. Holzäpfel, T. Baumgratz, C. Hempel, P. Jurcevic, I. Dhand, A. S. Buyskikh, A. J. Daley, M. Cramer, M. B. Plenio, R. Blatt, and C. F. Roos. "Efficient tomography of a quantum many-body system". Nature Physics **13**, 1158–1162 (2017).

[9] Joseph Emerson, Robert Alicki, and Karol Zyczkowski. "Scalable noise estimation with random unitary operators". Journal of Optics B: Quantum and Semiclassical Optics **7**, S347 (2005).

[10] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. "Randomized benchmarking of quantum gates". Physical Review A **77**, 012307 (2008).

[11] Alexander Erhard, Joel J. Wallman, Lukas Postler, Michael Meth, Roman Stricker, Esteban A. Martinez, Philipp Schindler, Thomas Monz, Joseph Emerson, and Rainer Blatt. "Characterizing large-scale quantum computers via cycle benchmarking". Nature Communications **10**, 5347 (2019).

[12] Timothy J. Proctor, Arnaud Carignan-Dugas, Kenneth Rudinger, Erik Nielsen, Robin Blume-Kohout, and Kevin Young. "Direct Randomized Benchmarking for Multiqubit Devices". Physical Review Letters **123**, 030503 (2019).

[13] Timothy Proctor, Stefan Seritan, Kenneth Rudinger, Erik Nielsen, Robin Blume-Kohout, and Kevin Young. "Scalable Randomized Benchmarking of Quantum Computers Using Mirror Circuits". Physical Review Letters **129**, 150502 (2022).

[14] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. "Characterizing quantum supremacy in near-term devices". Nature Physics **14**, 595–600 (2018).

[15] Jahan Claes, Eleanor Rieffel, and Zhihui Wang. "Character Randomized Benchmarking for Non-Multiplicity-Free Groups With Applications to Subspace, Leakage, and Matchgate Randomized Benchmarking". PRX Quantum **2**, 010351 (2021).

[16] Jonas Helsen, Sepehr Nezami, Matthew Reagor, and Michael Walter. "Matchgate benchmarking: Scalable benchmarking of a continuous family of many-qubit gates". Quantum **6**, 657 (2022).

[17] John Preskill. "Quantum Computing in the NISQ era and beyond". Quantum **2**, 79 (2018).

[18] Anders Sørensen and Klaus Mølmer. "Quantum computation with ions in thermal motion". Physical Review Letters **82**, 1971 (1999).

[19] Klaus Mølmer and Anders Sørensen. "Multiparticle Entanglement of Hot Trapped Ions". Physical Review Letters **82**, 1835–1838 (1999).

[20] Esteban A. Martinez, Thomas Monz, Daniel Nigg, Philipp Schindler, and Rainer Blatt.

"Compiling quantum algorithms for architectures with multi-qubit gates". New Journal of Physics **18**, 063029 (2016).

[21] V. Nebendahl, H. Häffner, and C. F. Roos. "Optimal control of entangling operations for trapped-ion quantum computing". Physical Review A **79**, 012312 (2009).

[22] Jordan Hines, Marie Lu, Ravi K. Naik, Akel Hashim, Jean-Loup Ville, Brad Mitchell, John Mark Kriekebaum, David I. Santiago, Stefan Seritan, Erik Nielsen, Robin Blume-Kohout, Kevin Young, Irfan Siddiqi, Birgitta Whaley, and Timothy Proctor. "Demonstrating scalable randomized benchmarking of universal gate sets" (2022). arXiv:2207.07272.

[23] T. Chasseur and F. K. Wilhelm. "Complete randomized benchmarking protocol accounting for leakage errors". Physical Review A **92**, 042333 (2015).

[24] Tobias Chasseur, Daniel M. Reich, Christiane P. Koch, and Frank K. Wilhelm. "Hybrid benchmarking of arbitrary quantum gates". Physical Review A **95**, 062335 (2017).

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

[26] Ryan Shaffer, Eli Megidish, Joseph Broz, Wei-Ting Chen, and Hartmut Häffner. "Practical verification protocols for analog quantum simulators". npj Quantum Information **7**, 1–12 (2021).

[27] Susan M. Clark, Daniel Lobser, Melissa C. Revelle, Christopher G. Yale, David Bossert, Ashlyn D. Burch, Matthew N. Chow, Craig W. Hogle, Megan Ivory, Jessica Pehr, Bradley Salzbrenner, Daniel Stick, William Sweatt, Joshua M. Wilson, Edward Winrow, and Peter Maunz. "Engineering the Quantum Scientific Computing Open User Testbed". IEEE Transactions on Quantum Engineering **2**, 1–32 (2021).

[28] IBM. "IBM Quantum". url: `https://quantum-computing.ibm.com/`.

[29] Aram W. Harrow and Richard A. Low. "Random quantum circuits are approximate 2-designs". Communications in Mathematical Physics **291**, 257–302 (2009).

[30] Alexander M. Dalzell, Nicholas Hunter-Jones, and Fernando G. S. L. Brandão. "Random quantum circuits transform local noise into global white noise" (2021). arXiv:2111.14907.

[31] Kristine Boone, Arnaud Carignan-Dugas, Joel J. Wallman, and Joseph Emerson. "Randomized benchmarking under different gate sets". Physical Review A **99**, 032329 (2019).

[32] Markus Heinrich, Martin Kliesch, and Ingo Roth. "General guarantees for randomized benchmarking with random quantum circuits" (2022). arXiv:2212.06181.

[33] Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T. Sornborger, and Patrick J. Coles. "Quantum-assisted quantum compiling". Quantum **3**, 140 (2019).

[34] Kunal Sharma, Sumeet Khatri, M Cerezo, and Patrick J Coles. "Noise resilience of variational quantum compiling". New Journal of Physics **22**, 043006 (2020).

[35] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". Biometrika **57**, 97–109 (1970).

[36] Michael A Nielsen. "A simple formula for the average gate fidelity of a quantum dynamical operation". Physics Letters A **303**, 249–252 (2002).

[37] R. Islam, W. C. Campbell, T. Choi, S. M. Clark, C. W. S. Conover, S. Debnath, E. E. Edwards, B. Fields, D. Hayes, D. Hucul, I. V. Inlek, K. G. Johnson, S. Korenblit, A. Lee, K. W. Lee, T. A. Manning, D. N. Matsukevich, J. Mizrahi, Q. Quraishi, C. Senko, J. Smith, and C. Monroe. "Beat note stabilization of mode-locked lasers for quantum information processing". Optics Letters **39**, 3238 (2014).

[38] D. Hayes, D. N. Matsukevich, P. Maunz, D. Hucul, Q. Quraishi, S. Olmschenk, W. Campbell, J. Mizrahi, C. Senko, and C. Monroe. "Entanglement of Atomic Qubits Using an Optical Frequency Comb". Physical Review Letters **104**, 140501 (2010).

[39] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe. "Demonstration of a small programmable quantum computer with atomic qubits". Nature **536**, 63–66 (2016).

[40] Kenneth R. Brown, Aram W. Harrow, and Isaac L. Chuang. "Arbitrarily accurate composite pulse sequences". Physical Review A **70**, 052318 (2004).

[41] P. J. Lee, K. A. Brickman, L. Deslauriers, P. C. Haljan, L. M. Duan, and C. Monroe. "Phase control of trapped ion quantum gates". Journal of Optics B: Quantum and Semiclassical Optics **7**, S371 (2005).

[42] Ryan Shaffer (2022). code: rmshaffer/stoq-compiler v0.2.0.

[43] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. Divincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. "Elementary gates for quantum computation". Physical Review A **52**, 3457 (1995).

[44] George Cybenko. "Reducing quantum computations to elementary unitary operations". Computing in Science and Engineering **3**, 27–32 (2001).

[45] Aram W. Harrow, Benjamin Recht, and Isaac L. Chuang. "Efficient discrete approximations of quantum gates". Journal of Mathematical Physics **43**, 4445–4451 (2002).

[46] Ali Javadiabhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T. Chong, and Margaret Martonosi. "ScaffCC: Scalable compilation and analysis of quantum programs". Parallel Computing **45**, 2–17 (2015).

[47] A. Kitaev. "Quantum computations: algorithms and error correction". Russian Mathematical Surveys **52**, 1191–1249 (1997).

[48] Eric Schkufza, Rahul Sharma, and Alex Aiken. "Stochastic superoptimization". In Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems. Pages 305–316. New York (2013). ACM Press.

[49] Joel J. Wallman and Joseph Emerson. "Noise tailoring for scalable quantum computation via randomized compiling". Physical Review A **94**, 52325 (2016).

[50] Naomichi Hatano and Masuo Suzuki. "Finding Exponential Product Formulas of Higher Orders". In Arnab Das and Bikas Chakrabarti, editors, Quantum Annealing and Other Optimization Methods. Chapter 2, pages 37–68. Springer, Berlin (2005).

[51] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". Quantum **3**, 163 (2019).

[52] Andrew M. Childs, Aaron Ostrander, and Yuan Su. "Faster quantum simulation by randomization". Quantum **3**, 182 (2019).

[53] Earl Campbell. "Random Compiler for Fast Hamiltonian Simulation". Physical Review Letters **123**, 070503 (2019).

[54] Yingkai Ouyang, David R. White, and Earl T. Campbell. "Compilation by stochastic Hamiltonian sparsification". Quantum **4**, 235 (2020).

[55] Francesco Mezzadri. "How to generate random matrices from the classical compact groups" (2006). arXiv:math-ph/0609050.

[56] E. Knill. "Approximation by Quantum Circuits" (1995). arXiv:quant-ph/9508006.

[57] David Poulin, Angie Qarry, Rolando Somma, and Frank Verstraete. "Quantum simulation of time-dependent Hamiltonians and the convenient illusion of Hilbert space". Physical Review Letters **106**, 170501 (2011).

A Derivation of RAV fidelity estimate

In this appendix, we derive the formula for the approximate fidelity of a RAV sequence on an n-qubit system. We start with the XEB fidelity estimate in Equation 1, where $P(x)$ represents the classically-computed ideal output probability distribution for the sequence, $Q(x)$ is the observed sample probability of obtaining measurement result x, and $N = 2^n$ is the dimension of the system. The RAV sequence is constructed to return nearly all of the population to the initial state x_0. Therefore, we let $P(x_0) = 1 - \epsilon$ for some small $\epsilon \ll 1$, which represents the approximation error of the inversion sequence. As a further simplification, we assume that the remaining probability is spread evenly among the remaining states, such that $P(x) = \frac{\epsilon}{N-1}$ for each $x \neq x_0$.

We can then derive the RAV fidelity from the XEB fidelity formula as follows:

$$\hat{F}_{\text{RAV}} = \frac{\sum_x P(x)Q(x) - \frac{1}{N}}{\sum_x P(x)^2 - \frac{1}{N}} \tag{11}$$

$$= \frac{P(x_0)Q(x_0) + \sum_{x \neq x_0} P(x)Q(x) - \frac{1}{N}}{P(x_0)^2 + \sum_{x \neq x_0} P(x)^2 - \frac{1}{N}} \tag{12}$$

$$= \frac{(1-\epsilon)Q(x_0) + \frac{\epsilon}{N-1}(1 - Q(x_0)) - \frac{1}{N}}{(1-\epsilon)^2 + \frac{1}{N-1}\epsilon^2 - \frac{1}{N}} \tag{13}$$

$$= \frac{NQ(x_0) - 1}{N-1} + \epsilon \, \frac{N(NQ(x_0) - 1)}{(N-1)^2} + \epsilon^2 \, \frac{N^2(NQ(x_0) - 1)}{(N-1)^3} + \cdots \tag{14}$$

$$= \frac{NQ(x_0) - 1}{N-1} \left[1 + \frac{N\epsilon}{N-1} + \left(\frac{N\epsilon}{N-1} \right)^2 + \cdots \right] \tag{15}$$

where in the next-to-last step we have performed a Taylor expansion around $\epsilon = 0$. Then, in the final step we note that the expression inside the square brackets is just a geometric series in $\frac{N\epsilon}{N-1}$, and therefore we have

$$\hat{F}_{\text{RAV}} = \frac{NQ(x_0) - 1}{N-1} \left[\frac{1}{1 - \frac{N\epsilon}{N-1}} \right] \tag{16}$$

$$= \frac{Q(x_0) - \frac{1}{N}}{(1-\epsilon) - \frac{1}{N}} \tag{17}$$

which, using the fact that $P(x_0) = 1 - \epsilon$, becomes:

$$\hat{F}_{\text{RAV}} = \frac{Q(x_0) - \frac{1}{N}}{P(x_0) - \frac{1}{N}} \tag{18}$$

B Derivation of variance in RAV and XEB fidelity estimates

In this appendix, we derive formulas for the variance of the fidelity estimates resulting from K independent shots of a single RAV or XEB circuit. We start by assuming that the errors in our device are purely depolarizing. Under this assumption, we can represent the output state of any n-qubit circuit as a mixture of the circuit's ideal output state $|\psi\rangle\langle\psi|$ and the maximally-mixed state $\frac{1}{N}I$ (where $N = 2^n$), where $\lambda \in [0, 1]$ is the fraction to which the output state is depolarized:

$$\rho_\lambda = (1 - \lambda)|\psi\rangle\langle\psi| + \frac{\lambda}{N} I \tag{19}$$

We can then define $P(x)$ and $Q_\lambda(x)$ as the probabilities of measuring outcome x when measuring the states $|\psi\rangle\langle\psi|$ and ρ_λ, respectively, as:

$$P(x) = \langle x|\psi\rangle\langle\psi|x\rangle = \left|\langle x|\psi\rangle\right|^2 \tag{20}$$

$$Q_\lambda(x) = \langle x|\rho_\lambda|x\rangle = (1 - \lambda)P(x) + \frac{\lambda}{N} \tag{21}$$

We can restate Equation 1 and Equation 2 to define the fidelity estimates $\hat{F}_{\text{RAV}}$ and $\hat{F}_{\text{XEB}}$ in terms of $P(x)$ and $Q_\lambda(x)$ as follows:

$$\hat{F}_{\text{RAV}} = \frac{Q_\lambda(x_0) - \frac{1}{N}}{P(x_0) - \frac{1}{N}} \tag{22}$$

$$\hat{F}_{\text{XEB}} = \frac{\sum_x P(x)Q_\lambda(x) - \frac{1}{N}}{\sum_x P(x)^2 - \frac{1}{N}} \tag{23}$$

Now, to calculate the expected variance of our these fidelity estimates, we first need to determine their distribution. To do this, we let $\mathbf{Q}_{\lambda,x} \sim \text{Multinomial}(K, N, Q_\lambda(x))$ be a random variable representing the number of times outcome x is observed when taking K independent shots of a RAV or XEB circuit, where $Q_\lambda(x)$ represents the "true" experimental probability distribution of each of N possible outcomes when measuring the state ρ_λ.

By the properties of the multinomial distribution, then, the variance of $\mathbf{Q}_{\lambda,x}$ is:

$$\text{Var}[\mathbf{Q}_{\lambda,x}] = KQ_\lambda(x)[1 - Q_\lambda(x)] \tag{24}$$

$$= K\left[(1-\lambda)P(x) + \frac{\lambda}{N}\right]\left[1 - (1-\lambda)P(x) - \frac{\lambda}{N}\right]. \tag{25}$$

We can now construct random variables corresponding to measurements of $\hat{F}_{\text{RAV}}$ and $\hat{F}_{\text{XEB}}$ by replacing $Q_\lambda(x)$ in Equation 22 and Equation 23 with the scaled random variable $\frac{1}{K}\mathbf{Q}_{\lambda,x}$, which represents the sample probability of observing outcome x when taking K independent shots:

$$\hat{\mathbf{F}}_{\text{RAV}} = \frac{\frac{1}{K}\mathbf{Q}_{\lambda,x_0} - \frac{1}{N}}{P(x_0) - \frac{1}{N}} \qquad \hat{\mathbf{F}}_{\text{XEB}} = \frac{\sum_x P(x)\frac{1}{K}\mathbf{Q}_{\lambda,x} - \frac{1}{N}}{\sum_x P(x)^2 - \frac{1}{N}} \tag{26}$$

Once we have done this, calculating the variance of these random variables is straightforward algebra:

$$\text{Var}[\hat{\mathbf{F}}_{\text{RAV}}] = \frac{1}{K^2}\left(\frac{1}{P(x_0) - \frac{1}{N}}\right)^2 \text{Var}[\mathbf{Q}_{\lambda,x_0}] \tag{27}$$

$$= \frac{1}{K}\left(\frac{1}{P(x_0) - \frac{1}{N}}\right)^2 \left[(1-\lambda)P(x_0) + \frac{\lambda}{N}\right]\left[1 - (1-\lambda)P(x_0) - \frac{\lambda}{N}\right] \tag{28}$$

$$\text{Var}[\hat{\mathbf{F}}_{\text{XEB}}] = \text{Var}\left[\frac{\frac{1}{K}\sum_x P(x)\mathbf{Q}_{\lambda,x} - \frac{1}{N}}{\sum_x P(x)^2 - \frac{1}{N}}\right] \tag{29}$$

$$= \frac{1}{K^2}\frac{\sum_x P(x)^2 \text{ Var}[\mathbf{Q}_{\lambda,x}]}{\left(\sum_x P(x)^2 - \frac{1}{N}\right)^2} \tag{30}$$

$$= \frac{1}{K^2}\left(\frac{1}{\sum_x P(x)^2 - \frac{1}{N}}\right)^2 \sum_x P(x)^2 \; K\left[(1-\lambda)P(x) + \frac{\lambda}{N}\right]\left[1 - (1-\lambda)P(x) - \frac{\lambda}{N}\right] \tag{31}$$

$$= \frac{1}{K}\left(\frac{1}{\sum_x P(x)^2 - \frac{1}{N}}\right)^2 \left[\left(\frac{\lambda}{N}\right)\left(1 - \frac{\lambda}{N}\right)\sum_x P(x)^2 + (1-\lambda)\left(1 - \frac{2\lambda}{N}\right)\sum_x P(x)^3 \right.$$
$$\left. - (1-\lambda)^2 \sum_x P(x)^4\right] \tag{32}$$

The above formulas are sufficient to predict the variance in fidelity measurements for a particular circuit, assuming we know the ideal probabilities $P(x)$ of measuring the circuit output. But of course, these probabilities will be different for every circuit. To make more general predictions, we need to make additional assumptions.

For RAV, the sequences have been explicitly constructed such that most of the population is returned to the initial state x_0. Therefore, we assume that $P(x_0) \approx 1-\epsilon$ for some small $\epsilon \ll 1$, which represents the approximation error of the inversion sequence. Substituting this into Equation 28 gives

$$\mathrm{Var}[\hat{\mathbf{F}}_{\mathrm{RAV}}] \approx \frac{1}{K}\left(\frac{1}{(1-\epsilon)-\frac{1}{N}}\right)^2\left[(1-\lambda)(1-\epsilon)+\frac{\lambda}{N}\right]\left[1-(1-\lambda)(1-\epsilon)-\frac{\lambda}{N}\right]. \tag{33}$$

For XEB, it is known that for ensembles of random circuits of large-enough depth on systems with tens of qubits, the distribution of ideal output state probabilities can be well-approximated by a Porter-Thomas distribution [14], in which the probabilities follow an exponential distribution $\Pr(Np) = Ne^{-Np}$. By the properties of the exponential distribution, then, we have $\sum_x P(x)^k \approx \frac{1}{k}$, and substituting this into Equation 32 gives

$$\mathrm{Var}[\hat{\mathbf{F}}_{\mathrm{XEB}}] \approx \frac{1}{K}\left(\frac{1}{\frac{1}{2}-\frac{1}{N}}\right)^2\left[\frac{1}{2}\left(\frac{\lambda}{N}\right)\left(1-\frac{\lambda}{N}\right)+\frac{1}{3}(1-\lambda)\left(1-\frac{2\lambda}{N}\right)-\frac{1}{4}(1-\lambda)^2\right]. \tag{34}$$

It is important to note that in Figure 4 we are working with simulations of sequences with $n \leq 8$ qubits and $10 \leq m \leq 30$ layers, and so the Porter-Thomas assumption is not necessarily valid in this regime. This is the most likely explanation for the systematic discrepancies between this estimate of the XEB variance and the observed XEB variance in our simulations. We also note that the convergence to the Porter-Thomas distribution applies only to ensembles of circuits, since a fixed circuit will have a particular distribution that does not "converge" toward anything. The expression in Equation 34 should therefore be thought of as an "expected" variance over all XEB circuits, rather than a variance for a particular fixed circuit.

C Approximate unitary compilation via stochastic search (STOQ)

In this appendix, we supplement the main text with additional implementation details of STOQ and examples of its use. We report results of applying STOQ to Hamiltonian time-evolution unitaries, where we compare its performance to existing methods on various metrics. We also demonstrate the use of STOQ to approximately compile gate sequences for randomly-generated unitaries, although as one would expect, its performance scales poorly with the required circuit depth. We conclude with additional discussion of STOQ, including its features and limitations.

C.1 Background

A critical prerequisite to executing any algorithm on a physical quantum computer is the process commonly known as quantum compilation. One of the primary tasks of quantum compilation is the conversion of a target unitary operation into a sequence of quantum gates that are native to the physical device being used [43, 44, 45, 46]. Because unitary operators belong to a continuous space, such compilation in general results in gate sequences which are only approximately equivalent to the target unitary. For example, one of the earliest quantum compilation techniques, the Solovay-Kitaev method [47], compiles gate sequences that differ from the target unitary by an amount that can be made as small as desired.

Traditional compilation, both in the classical and quantum realms, is most often a deterministic process, using rules and heuristics to efficiently synthesize a desired program from the native assembly instructions (in classical compilation) or native physical gates (in quantum compilation). But in some cases, adding stochasticity to the compilation process has been shown to produce advantages in the resulting program. In classical compilation, a technique known as stochastic superoptimization [48] has been shown in certain cases to produce significantly shorter programs than the best-in-class compilers and optimizers. In quantum compilation, techniques such as randomized compiling [49] have been demonstrated to improve noise resilience by randomizing errors that occur during program execution.

In the field of quantum compilation, special attention has been paid to compilation of unitaries which result from the time-evolution of physically-realizable Hamiltonians. The compiled sequences in these cases can be executed to perform what is known as "Hamiltonian simulation", or more broadly, "quantum simulation". Such approaches are of special interest in fields such as quantum chemistry, where it is desirable to use a quantum computer to simulate the dynamics of physical systems. Common approaches to this problem include product formula techniques such as the Suzuki-Trotter decomposition [50] and qubitization [51], which deterministically compile the time-evolution unitary for a given Hamiltonian into a sequence of quantum gates.

Approaches involving stochasticity have recently been shown to be advantageous in some cases. Adding randomization to the Suzuki-Trotter decomposition [52] creates approximate compilations that are better both theoretically and empirically. A stochastic compilation protocol known as QDRIFT [53], where gate probabilities are weighted according to the strength of each term in the Hamiltonian rather than using a product formula directly, has been shown to produce much more efficient compilations in many cases. An interpolation of these two methods [54] has also been proposed, which takes some of the advantages of each method. The efficiency of these compilation methods is generally independent of system size when applied to problems involving sparse Hamiltonians.

However, these specialized methods cannot be applied to general-purpose compilation tasks, which is where we focus specifically here. In Section 2.2 of the main text, a stochastic approximate quantum unitary compilation procedure, abbreviated as STOQ, is described in detail. The STOQ protocol was originally developed as part of a verification scheme for analog quantum simulators called *randomized analog verification* (RAV) [26]. And in this work, an adaptation of RAV for gate-based quantum devices is summarized in Section 2.1, demonstrated numerically in Section 3.1, and demonstrated experimentally in Section 3.2. This gate-based version of RAV also uses STOQ to compile the approximate inversion portion of each sequence.

C.2 Additional implementation details

This section fills in a few important details of the STOQ protocol implementation outlined in Section 2.2, specifically referring to the pseudocode representation in Figure 6.

The compiled sequence is stored in the `sequence` variable, which is initially empty. The `RandomChange` function returns a modified sequence on each iteration, either by adding a randomly-drawn instruction to the sequence from the parameterized instruction set `G` with randomly-generated parameter values, or by removing an instruction from the sequence. The `Prod` function calculates the unitary that represents the product of all of the operations in the sequence, and the `Cost` function is implemented as described in Equation 7.

The variable `beta` is used as an annealing parameter for the compilation process. The function `IncreaseBeta` returns a slightly increased value of `beta` on each iteration. Defining the annealing parameter as $\beta =$ `beta` and the cost difference of such a proposed change as $\Delta =$ `new_cost` $-$ `cost`, the `Accept` function calculates the probability of accepting a proposed change as

$$P_{\text{accept}} = \begin{cases} e^{-\beta\Delta} & \Delta > 0 \\ 1 & \Delta \leq 0. \end{cases} \tag{35}$$

The probability of accepting "bad" proposed changes where the cost increases (i.e., where $\Delta > 0$) approaches zero as β increases.

C.3 Compilation of time-evolution unitaries

To demonstrate one possible (although not necessarily useful) application of STOQ, we choose an Ising-type Hamiltonian with nearest-neighbor coupling and transverse field

$$H = \sum_{<i,j>} J_{ij}\sigma_x^{(i)}\sigma_x^{(j)} + \sum_i h_i\sigma_y^{(i)} \tag{36}$$

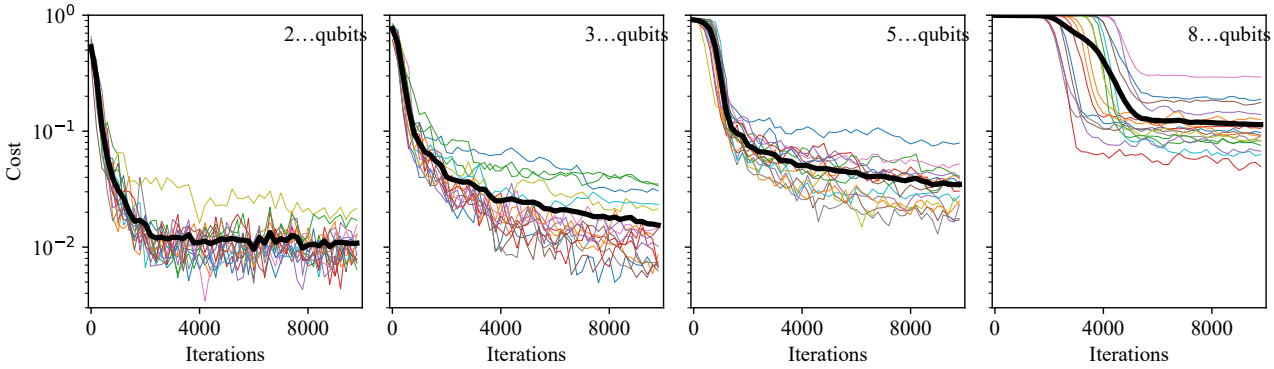


Figure 10: Compilation via STOQ for two-qubit, three-qubit, five-qubit, and eight-qubit versions of the time-evolution unitary from Equation 37. Each of the 16 thin curves shows the value of the cost function from Equation 7 during a single compilation using 10,000 iterations. The thick curve is the average of all runs.

n	J_{12}	J_{23}	J_{34}	J_{45}	J_{56}	J_{67}	J_{78}	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
2	1.27							1.54	1.19						
3	1.81	1.27						1.54	1.19	0.53					
5	1.20	1.40	1.60	1.80				1.60	1.30	1.00	0.70	0.40			
8	1.20	1.30	1.40	1.50	1.60	1.70	1.80	1.40	1.10	0.80	1.00	1.20	1.50	1.70	1.30

Table 1: Coefficients used for application of STOQ to the n-qubit Ising model Hamiltonian in Equation 36. Values are energies in kHz where $\hbar = 1$.

where the coefficients J_{ij} and h_i are energies with particular values for each system size, as shown in Table 1.

We then define the time-evolution unitary as $U_t(\tau) = e^{iH\tau}$, where we choose units such that $\hbar = 1$, and we concretely choose $\tau = 0.5$ ms, such that

$$U = U_t(0.5 \text{ ms}) = e^{iH(0.5 \text{ ms})} \tag{37}$$

is the target unitary for compilation.

To apply STOQ, we need also to choose a parameterized instruction set G from which to approximately compile a sequence. In a physical device, it is often the case that the dynamics are implemented such that each term in H can be individually controlled. To define G for such a device, we express the Hamiltonian as $H = \sum_k H_k$, where each H_k is one of the $\sigma_x\sigma_x$ or σ_y terms from Equation 36, and choose

$$G = \bigcup_k \left\{ e^{iH_k t} \right\} \quad -\epsilon\tau \leq t \leq \epsilon\tau \tag{38}$$

where the allowed range for t is chosen such that the duration of each instruction is relatively short in comparison to the timescale of the dynamics of H. (In this demonstration we use $\epsilon = 0.2$.) Negative times correspond to reversing the sign of the coefficient of a given term. (We note that for a general Hamiltonian, it is unlikely that each H_k term is part of the native gate set of the device. In this case, G should instead represent the interactions which are implemented natively on the device.)

We then apply STOQ to compile many sequences that approximately implement U, using two-qubit, three-qubit, five-qubit, and eight-qubit versions of the corresponding Hamiltonian. Figure 10 reports the cost for 16 such compilations as a function of the number of iterations. (Each run of 10,000 iterations for the five-qubit system takes around 15 minutes to complete on a typical desktop computer.) We observe that the stochastic search process rapidly reduces the cost at first before noticeably leveling off. For the two-qubit and three-qubit systems, this cost approaches a limit near 10^{-2} after 10,000 iterations. For the larger systems, the final average cost is higher, although even for the eight-qubit system, the final cost reaches a value below 10^{-1} for some compilations.

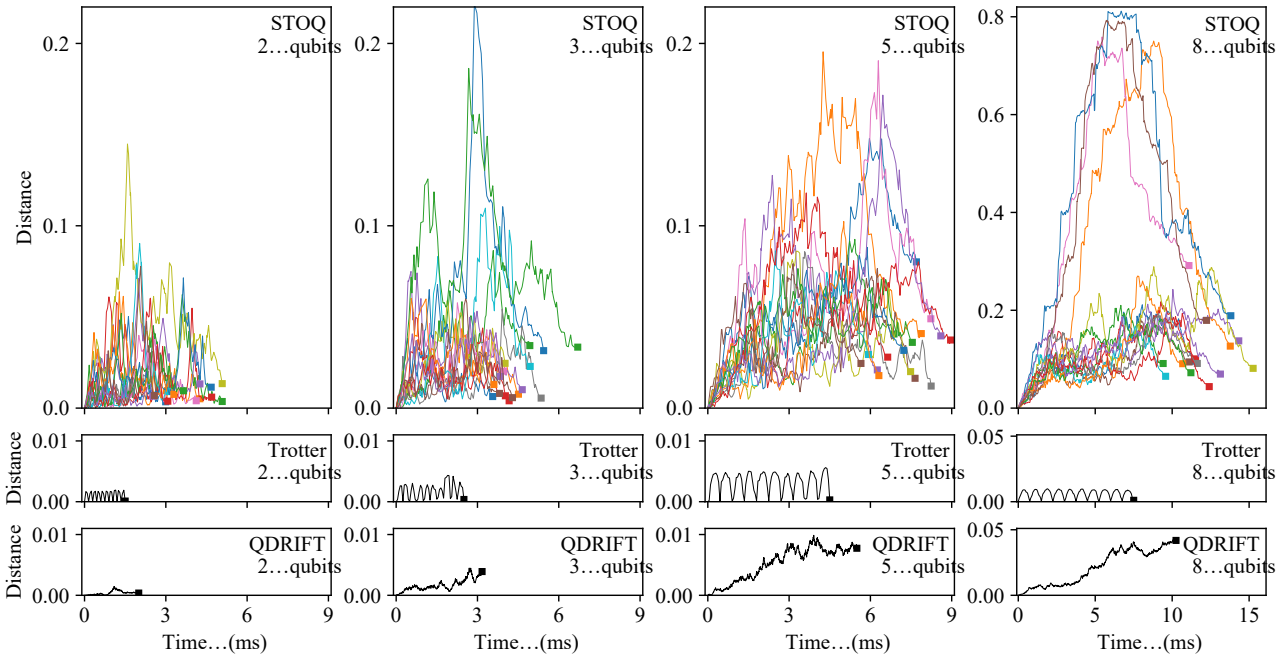


Figure 11: Distance from ideal path to compiled path, as defined in Equation 39, for the time-evolution unitary from Equation 37, illustrating "how different" a particular compilation is from the path that would be followed by the ideal time evolution of the full Hamiltonian. The distance from the horizontal axis quantifies the deviation in the system state from the ideal simulation when executing the specified compilation. The ideal simulation would be a flat curve along the horizontal axis from $\tau = 0.0$ ms to $\tau = 0.5$ ms. Results are shown for 2-qubit, 3-qubit, 5-qubit, and 8-qubit implementations of the Ising model Hamiltonian from Equation 36. Each curve represents the execution of one compiled sequence. Filled squares are used to plot the overall running time of the compiled sequence and final cost of each compilation. Top row depicts the execution of 16 independent STOQ compilations, each using 10,000 iterations. Each curve corresponds to a curve of the same color in Figure 10. Middle row depicts the execution of a typical randomized Suzuki-Trotter compilation using 10 steps. Bottom row depicts the execution of a typical QDRIFT compilation using 1,000 repetitions. The horizontal axis of each plot represents the execution time of the compiled circuits.

	Ideal	Trotter	QDRIFT	STOQ
(a) Execution time (ms)	0.50	4.50	5.50	7.32
(b) Average distance	—	0.0032	0.0053	0.0469
(c) Maximum distance	—	0.0056	0.0099	0.1133
(d) Final cost	—	0.0003	0.0077	0.0328

Table 2: Statistics resulting from various compilations of the five-qubit time-evolution unitary from Equation 37, where the ideal evolution occurs for $\tau = 0.5$ ms. For each of the compilation techniques, means are listed for each of the following quantities: (a) total execution time of the compiled sequence, (b) average distance $\sum_{m=1}^{M} d_m$, (c) maximum distance $\max_{m\in[1,M]} d_m$, and (d) final cost d_M. Corresponds to five-qubit plots in Figure 11.

To compare STOQ to existing compilation techniques, we also compile sequences to approximately implement U using the randomized Suzuki-Trotter decomposition [52] and the QDRIFT stochastic compilation protocol [53]. STOQ is designed to create more randomness in the resulting path taken through state space. To compare these paths quantitatively, we choose to compare the various methods to an ideal version where H is directly implemented for time τ. We define the *ideal path* as the path taken by this ideal time evolution, and we define the *compiled path* as the path taken by the compiled sequence, which we represent as a sequence of instructions $\{G_1, \ldots, G_M\}$. We then calculate the path distance d_m from the ideal path to step m of the compiled path, where $1 \leq m \leq M$, as

$$d_m = \min_{t\,\in\,[0,\tau]} D_{\text{HS}}\left(e^{iHt},\ G_m G_{m-1} \cdots G_1\right), \tag{39}$$

where D_{HS} is the distance metric defined in Equation 6. Thus d_m is the shortest distance from step m of the compiled path to any point in the ideal path.

Results for each compilation technique are shown in Figure 11, and statistics for the five-qubit example are displayed in Table 2. We observe that the STOQ compilations result in a significantly greater path distance from the ideal evolution than the other approaches, and that the total running time of the compiled sequence resulting from the various compilations is within a factor of two.

However, the final cost of the STOQ compilations is typically at least an order of magnitude larger than the compilations created using the randomized Suzuki-Trotter and QDRIFT techniques, both of which can reach arbitrarily low costs by increasing the number of steps. This implies that STOQ would not be a useful tool for applications that require high-fidelity compilations.

C.4 Compilation of random unitaries

In addition to being used for sparse or highly structured unitaries such as those generated from Hamiltonian time-evolution, the STOQ protocol can also be used to compile sequences that approximately implement purely random unitaries in terms of an arbitrary instruction set, without having any prior knowledge of the structure of the unitary. Such an application of STOQ is not necessarily useful in general, but is included here for illustrative purposes.

Figure 12 shows typical results of repeatedly using the STOQ protocol to compile sequences for random two-qubit, three-qubit, and five-qubit unitaries, chosen uniformly at random from the Haar measure [55], using a simple universal instruction set $G = \{R(\theta, \varphi), XX(\theta)\}$. $R(\theta, \varphi)$ is a parameterized single-qubit rotation

$$R(\theta, \varphi) = \begin{bmatrix} \cos\frac{\theta}{2} & -ie^{i\varphi}\sin\frac{\theta}{2} \\ ie^{-i\varphi}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{40}$$

with $0 \leq \theta < 2\pi$ and $0 \leq \varphi < 2\pi$. $XX(\theta)$ is a parameterized two-qubit entangling gate

$$XX(\theta) = \begin{bmatrix} \cos\theta & 0 & 0 & -i\sin\theta \\ 0 & \cos\theta & -i\sin\theta & 0 \\ 0 & -i\sin\theta & \cos\theta & 0 \\ -i\sin\theta & 0 & 0 & \cos\theta \end{bmatrix} \tag{41}$$

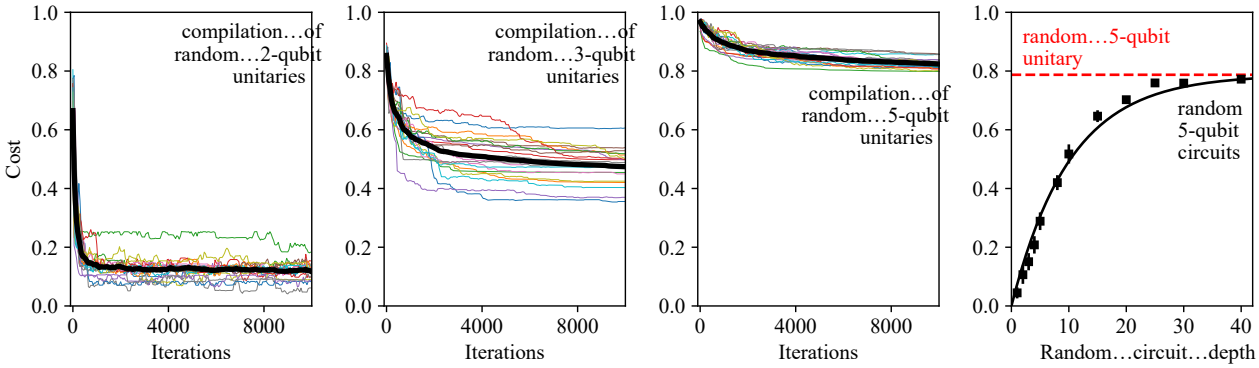


Figure 12: Compilation via STOQ of unitaries chosen uniformly at random from the Haar measure. The left three plots show the cost during the STOQ compilation process for randomly-generated 2-qubit, 3-qubit, and 5-qubit target unitaries. Each of the 20 thin curves shows the value of the cost function from Equation 7 during a single compilation using 10,000 iterations. The thick curve is the average of all runs. The rightmost plot shows the final cost of the STOQ compilation for target unitaries generated by creating random 5-qubit circuits of varying average circuit depth. Circuit depth is calculated as the total number of instructions divided by the number of qubits. Each point is the average of 20 compilations using 100,000 iterations. Error bars indicate standard error of the mean. The solid line is an exponential decay fit with one free parameter. The dashed line represents the average final cost of compiling a randomly-generated 5-qubit unitary. Note that one would expect a similar scaling with circuit depth for general quantum circuits, regardless of whether they are generated randomly.

with $0 \leq \theta < 2\pi$. We note that the instruction set G is a typical native gate set that can be implemented by trapped-ion quantum devices.

We observe that the final costs of compilation of these random unitaries are significantly larger than for compilation of the time-evolution unitaries discussed in Appendix C.3. In particular, the final cost is approximately 0.1 for two-qubit random unitaries, 0.5 for three-qubit random unitaries, and 0.8 for five-qubit random unitaries. This indicates that the quality of the approximation for such random unitary compilations scales poorly with system size. This is not surprising, since reaching the vast majority of states in the Hilbert space of a system requires circuits of depth which grows exponentially with the dimension of the Hilbert space [56, 57]. Nonetheless, the compilations generated by this method may be useful in scenarios where high-fidelity approximations are not required.

We also observe that the final cost of such random unitary compilations is relatively stable over a wide range of STOQ parameter values. Two primary parameters that can be adjusted in the STOQ algorithm in Figure 6 are the annealing rate $\Delta\beta$, which is used to increment β at each step inside the `IncreaseBeta` function, and the probability p_{append} that the search appends an instruction (as opposed to removing an instruction) at each step, which occurs inside the `RandomChange` function. For compilation of three-qubit random unitaries, and for values $\Delta\beta \in \{0.001, 0.01, 0.1, 0.5\}$ and $p_{\text{append}} \in \{0.2, 0.5, 0.8\}$, we find that the average final cost remains between 0.398 (for $\Delta\beta = 0.5$ and $p_{\text{append}} = 0.2$) and 0.448 (for $\Delta\beta = 0.001$ and $p_{\text{append}} = 0.5$), where each pair of parameter values is averaged over 32 compilations using 100,000 iterations each.

To provide insight into the low-fidelity approximations of random unitaries produced by STOQ, we consider the case of target unitaries generated by random circuits of varying depth. To do this, we generate random five-qubit circuits of average depth ranging from 1 to 40, where the average depth is calculated as the total number of instructions divided by the number of qubits. The rightmost plot in Figure 12 shows the final compilation cost after applying STOQ to generate an approximate compilation of the unitary corresponding to each random circuit. As might be expected, we observe that STOQ generates relatively high-fidelity approximations for shallow circuits, since such unitaries are known to be reachable with a fixed number of instructions. But as the circuit depth increases, the resulting unitaries begin to look more like random unitaries, and the final compilation cost approaches that of the randomly-generated five-qubit unitary discussed previously. Indeed, we expect a similar scaling with circuit depth for quantum circuits in general, regardless of whether they are randomly

generated, since the size of the reachable state space grows exponentially with the depth of the circuit.

C.5 Discussion

We note that because the STOQ protocol requires calculating the product of the compiled sequence during each iteration, the computational cost of each iteration grows exponentially in the system size n. Therefore, STOQ is unlikely to be useful for system sizes of more than around 10 qubits. In particular, for compilation of time-evolution unitaries, this clearly means that STOQ will be less computationally efficient when compared to compilation methods based on product formulas, which in general have a computational cost that depends only on the number of terms in the Hamiltonian and is independent of the system size.

We note that unitaries generated via time evolution of a Hamiltonian often benefit from the sparsity of the Hamiltonian. In general, an n-qubit Hamiltonian has 4^n coefficients when expressed in the basis of Pauli operators. For the five-qubit version of the Hamiltonian in Equation 36, only nine of these 1024 coefficients are non-zero. Sparsity in the Hamiltonian greatly limits the subspace of the full operator space that can be reached by via time evolution, which in turn makes compilation a more feasible task and allows techniques such as Suzuki-Trotter and QDRIFT to be highly efficient. Because the number of possible step proposals during each iteration of the STOQ search process is determined by the number of terms in the Hamiltonian, it is reasonable to infer that STOQ is similarly more effective when the problem structure contains such sparsity. This is further evidenced by the inability of the STOQ protocol to efficiently obtain low cost values when compiling sequences for random target unitaries, which are not sparse in general.

As demonstrated, STOQ has some features that are distinct from other methods. One notable feature is the capability of generating results with arbitrary gate sets, regardless of whether the gates are fixed or parameterized or whether the gate set is universal. In addition, repeated application of STOQ provides many independent approximate compilations of the same unitary, and each compilation creates a sequence that will cause the system state to traverse a different path in state space. As depicted in Figure 11, even stochastic techniques such as randomized Suzuki-Trotter or QDRIFT result in a compiled sequence that will cause the system state to follow very nearly the same path in state space as the deterministic version, whereas sequences generated by STOQ cause the system to traverse unique paths that can differ greatly from the ideal path and from each other.