

Extensive review of Benchmarking Quantum Computers

(July, 2022)

Quantum Computation: Introduction to algorithms and
implementation using QISKIT

Conducted by IISER Tirupati, Qkrishi

TABLE OF CONTENTS

ABSTRACT.....	2
AUTHORS.....	2
0. INTRODUCTION	3
0.1 WHAT IS BENCHMARKING?	3
0.2 WHAT ARE WE TRYING TO BENCHMARK IN QUANTUM COMPUTERS?.....	3
0.3 CURRENT STATE OF BENCHMARKING QUANTUM COMPUTERS.....	6
0.4 TYPE OF BENCHMARKS	6
1.QUANTUM STATE TOMOGRAPHY	8
1.1 INTRODUCTION	8
1.2 EXACT TOMOGRAPHY	8
1.3 SINGLE QUBIT TOMOGRAPHY.....	8
1.4 EXACT TOMOGRAPHY OF MULTIPLE QUBITS USING	10
1.5 ERRORS IN REAL TOMOGRAPHY	10
1.6 TYPES OF ERRORS	10
1.7 ERROR ANALYSIS.....	12
2. RANDOMIZED BENCHMARKING	15
2.1 INTRODUCTION	15
2.2 WHAT IS RANDOMIZED BENCHMARKING?	15
2.3 CLIFFORD GROUP	16
2.4 RB METHODS FOR ESTIMATING NOISE PARAMETERS.....	16
2.4 RB METHODS FOR OTHER GROUPS AND GATES.....	17
2.5 THEORETICAL BACKGROUND OF RB.....	18
2.6 THEORETICAL BACKGROUND OF THE CLIFFORD GROUP	20
3. QUANTUM VOLUME	21
3.1 INTRODUCTION	21
3.2 WHAT IS IT AND ITS IMPORTANCE	21
3.3 CALCULATION OF QUANTUM VOLUME.....	22
3.4 APPLICATION OF QUANTUM VOLUME.....	23
3.5 LIMITATION OF QUANTUM VOLUME	24
3.6 SUMMARY	24
4. CROSS ENTROPY BENCHMARKING	26
4.1 INTRODUCTION	26
4.2 CROSS ENTROPY BENCHMARKING.....	27
4.3 CLIFFORD XEB	28
5. ALGORITHMS BASED QUANTUM COMPUTER BENCHMARKING	30
5.1 INTRODUCTION	30
5.2 METHOD	31
5.3 BENCHMARK INDICATORS.....	34
5.4 INTERPRETATION	35
5.5 SUMMARY	37
CONCLUSION	38
BIBLIOGRAPHY.....	39
APPENDIX	41
RB CODE FOR SIMULATIONS.....	41

Abstract

The race to quantum advantage depends upon benchmarking. Systems benchmarks measure fundamental characteristics of a quantum computer. Such measures enable assessments of machine performance without regard to specific use cases and provide a clear record of progress. Here we have discussed various aspects of some of such benchmarking schemes. We have explored various techniques using different methods to Measure Power and performance of System.

Authors

Aiswarya M R, Ankush Kaushik, Athira Venu, Ayan Barui, Harsh Mehta, Hirak Ghosh, Shruti Jain, Soumik Samanta

0. Introduction

0.1 What is Benchmarking?

Benchmarking is how the performance & capabilities of a computing system is determined. One must choose the appropriate benchmark and metrics to extract meaningful results. Different benchmarks test the system in different ways and each individual metric may or may not be of interest. Some vital characteristics for Benchmarks:

- **Relevance:** Benchmarks should measure important features.
- **Representativeness:** Benchmark performance metrics should be broadly accepted by industry and academia.
- **Equity:** All systems should be fairly compared.
- **Repeatability:** Benchmark results should be verifiable.
- **Cost-effectiveness:** Benchmark tests should be economical.
- **Scalability:** Benchmark tests should measure from single server to multiple servers.
- **Transparency:** Benchmark metrics should be readily understandable.

Performance Metrics in Classical Computing:

- **Raw Performance of Individual Hardware Components**
- **Time to Solution-** Getting the job done as soon as possible
- **Throughput-** Number of jobs that the computer can satisfy simultaneously
- **Instruction Set Architecture-** Compatibility

0.2 What are we trying to Benchmark in Quantum Computers?

Benchmarking in quantum computing is more complicated than in other fields, where there is a wider range of hardware (from laser-controlled arrays of ions to microwave-controlled spinning electrons), fewer established guidelines, and additional complicating factors. The capacity of a quantum computer to perform meaningful computations is not just determined by the number of qubits, but by the quality of those qubits. Harnessing the power of quantum computing requires overcoming errors and other imperfections that are inevitable in all quantum computing hardware. Quantum Computer components fail in many different ways:

- Unintended bit-flips etc
- Systematic errors that add up coherently and unpredictably
- Drift in performance over (Operations drifting out of calibration)
- Complex & poorly understood “crosstalk” and integration failures in larger devices

In the near and medium-term, benchmarking quantum computers is all about quantifying all sources of error and understanding how this errors impact applications

Complexity of Error Problem

- Complexity of error model scales faster than the exponential gain of the quantum computer

For an n-qubit quantum computer:

- Quantum computing power **grows exponentially:** 2^n
- Quantum error modes **grow even faster*:** 2^{4n}

*This is assuming only Markovian errors on the time-scale of gates, otherwise the complexity is even worse

Performance Metrics in Quantum Computing:

Measurements Protocols- Some Single Qubit Metrics

- **Preparation fidelity:** ability to prepare a qubit in a reference state
- **Measurement fidelity:** ability to distinguish $|0\rangle$ and in $|1\rangle$ measurement.
- **Coherence time T1:** Usually time for $|1\rangle$ to decay into $|0\rangle$
- **Coherence time T2:** Time for superposition $|0\rangle + |1\rangle$ to lose phase coherence.
- **Logic gate fidelities.**
- **Logic gate times.**
- **Number of logic gates / decoherence time.**
- **How many operations you can do before things decohere.**

Measurements Protocols- Some Few-Qubit Metrics

- **Average gate error (for each qubits)**
- **Connectivity (which qubits connect to which qubits)**
- **Crosstalk (unwanted interaction between qubits)**
- **Time dependence AKA “non-Markovianity” (gates get worse later in calculation)**

- Drift of starting conditions, such as temperature, power of lasers, microwave pulse control parameters
- **Measurement / feed-forward time**
 - Relevant when the measurement of a qubit is used to decide what to do next ; an essential part of error correction
- **Circuit depth (Number of logic gates in series)**
 - Limited by coherence times, time for a gate to function, temperature change, etc.
 - Very demanding metric

Measurement Protocols- Some System-Level Metrics

- **Number of qubits**
 - Popular but necessarily meaningful
- **Quantum Volume**
- **Exemplar algorithms**
 - Shor's, Grover's, etc.
- **Time to solve**
- **Fidelity / quality of solution**
- **Comparison to “best” Classical computer**
 - Quantum Supremacy

0.3 Current State of Benchmarking Quantum Computers

	Systems benchmark ←						Application benchmark →
Origin	IBM		Sandia National Laboratories	UC Berkeley / Berkeley Lab	Atos	QED-C	Super.tech
Benchmark	Quantum Volume	Circuit Layer Ops / Second (CLOPS)	Mirror Circuits	Quantum LINPACK	Q-Score	App-Oriented Suite	SupermarQ Suite
Basis	Maximum size of square quantum circuits that can be implemented	Speed in executing layers of a parameterized model circuit	Executing quantum circuits forward and backward	Performance in a prerequisite task for linear algebra	Performance in solving a standard optimization problem	Executing common quantum algorithms / programs	Executing common quantum algorithms / programs plus error correction
Pros	<ul style="list-style-type: none"> ✓ Inclusive measure of performance ✓ Practical measure of noise ✓ Cannot be "gamed" with classical improvements 	<ul style="list-style-type: none"> ✓ Evaluates speed of whole-machine operations ✓ Covers quantum-classical latency 	<ul style="list-style-type: none"> ✓ Captures significant error sources outside of gate error rates ✓ The forward-backward "mirror" execution makes the benchmarks easily verifiable 	<ul style="list-style-type: none"> ✓ Predicts efficacy in scientific computing applications 	<ul style="list-style-type: none"> ✓ Predicts efficacy in optimization applications 	<ul style="list-style-type: none"> ✓ Targeted toward practical applications ✓ Evaluates whole-machine operations 	<ul style="list-style-type: none"> ✓ Targeted toward practical applications ✓ Evaluates whole-machine operations ✓ Scalable for post-supremacy testing
Cons	<ul style="list-style-type: none"> ✗ Non-square circuits can be predicted only directionally ✗ Applies only to near-term quantum computers 	<ul style="list-style-type: none"> ✗ May not measure speed in all applications ✗ No distinction between classical and quantum improvements 	<ul style="list-style-type: none"> ✗ Indirect measure of application performance 	<ul style="list-style-type: none"> ✗ Less useful for comparing computers that "pass" the test ✗ Restricted to linear algebra 	<ul style="list-style-type: none"> ✗ Restricted to optimization 	<ul style="list-style-type: none"> ✗ Applies only to near-term quantum computers ✗ No distinction between open and closed systems ✗ No distinction between classical and quantum improvements 	<ul style="list-style-type: none"> ✗ No distinction between classical and quantum improvements ✗ No distinction between open and closed systems

Source: IBM; Sandia National Laboratories; UC Berkeley; American Physical Society; Atos; Quantum Economic Development Consortium; Super.tech; BCG analysis.

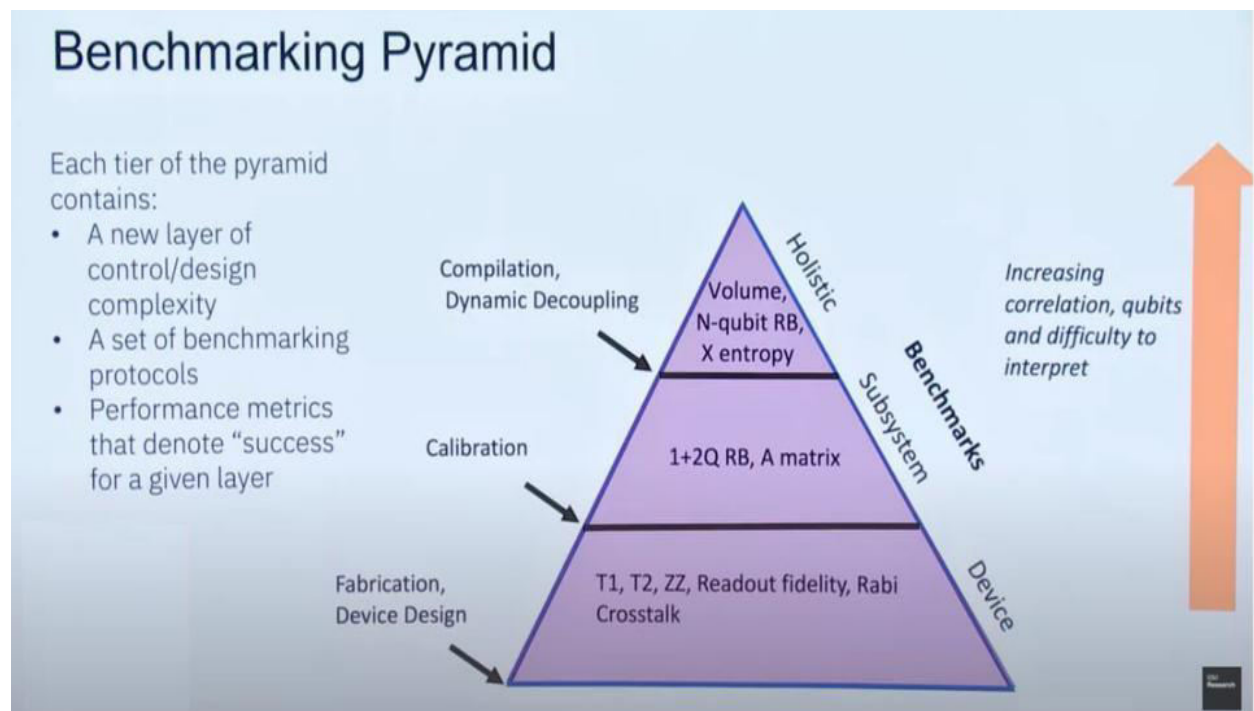
0.4 Type of Benchmarks

System Benchmark: Hardware Performance with such matrices

- **Scale (number of qubits):** Amount of information that can be encoded in quantum system
- **Quality:** Quality of circuits implemented in hardware with low operation errors
- **Speed:** The number of circuits that can run on hardware in given time

Application Benchmark: How well a specific quantum device work at specific task

- **LINPACK-** perform a key prerequisite computational task for linear algebra
- **Q-Score-** performance on a full optimization problem by assessing the maximum number of variables a quantum processor can optimize
- **SupermarQ-** applies techniques from classical benchmarking methodologies to the quantum domain.



We have a lot of low-level benchmarks for charactering few-qubit devices, or a few qubits at a time in a large device (This is how standard data is obtained)

- **Quantum State Tomography**
- **Randomized Benchmarking**

But most current methods do not scale too many qubits! There is a bunch of recent methods that have better scalability

- **Quantum Volume**
- **Cross Entropy Benchmarking**

There is some application-oriented benchmarks that test the performance of quantum computers on practically relevant tasks

- **Algorithms Based Benchmarking**

1. Quantum state Tomography

1.1 Introduction

Quantum Tomography characterises the complete quantum state of a particles a or particles through a series of measurements in different bases. For a characterisation of a classical object; the process involves a series of measurements, but measuring a single quantum particle in this way may perturb its sate, and thus makes its further investigation uninformative. Because of this reason, Quantum Tomography must be carried out in stages of a number of identical copies of the same state, and can never be successfully applied to a single unknown particle. A series of measurements on identical particle ensembles each allow a glimpse into a different aspect of a quantum state's reality. Each new type of measurement illuminates a new dimension of an unknown state; subjecting more identical copies of that state to a single type of measurement brings that particular observable into sharper relief.

1.2 Exact tomography

The goal of tomography is to reconstruct the density matrix of an unknown ensemble of particles through a series of measurements. In practice, this can never be performed exactly, as an infinite number of particles would be required to eliminate statistical error. If exact measurements were taken on infinite ensembles, each measurement would yield an exact probability of success, which could then be used to reconstruct a density matrix. Though unrealistic, it is highly illustrative to examine this exact tomography before seeing the more general treatment. Hence, this section will treat all measurements as yielding exact probabilities, and ignore all sources of error in those measurements.

1.3 Single Qubit Tomography

Although reconstructive tomography of any size system follows the same general procedure, beginning with tomography of a single qubit allows the visualization of each step using the Poincare sphere, in addition to providing a simpler mathematical introduction.

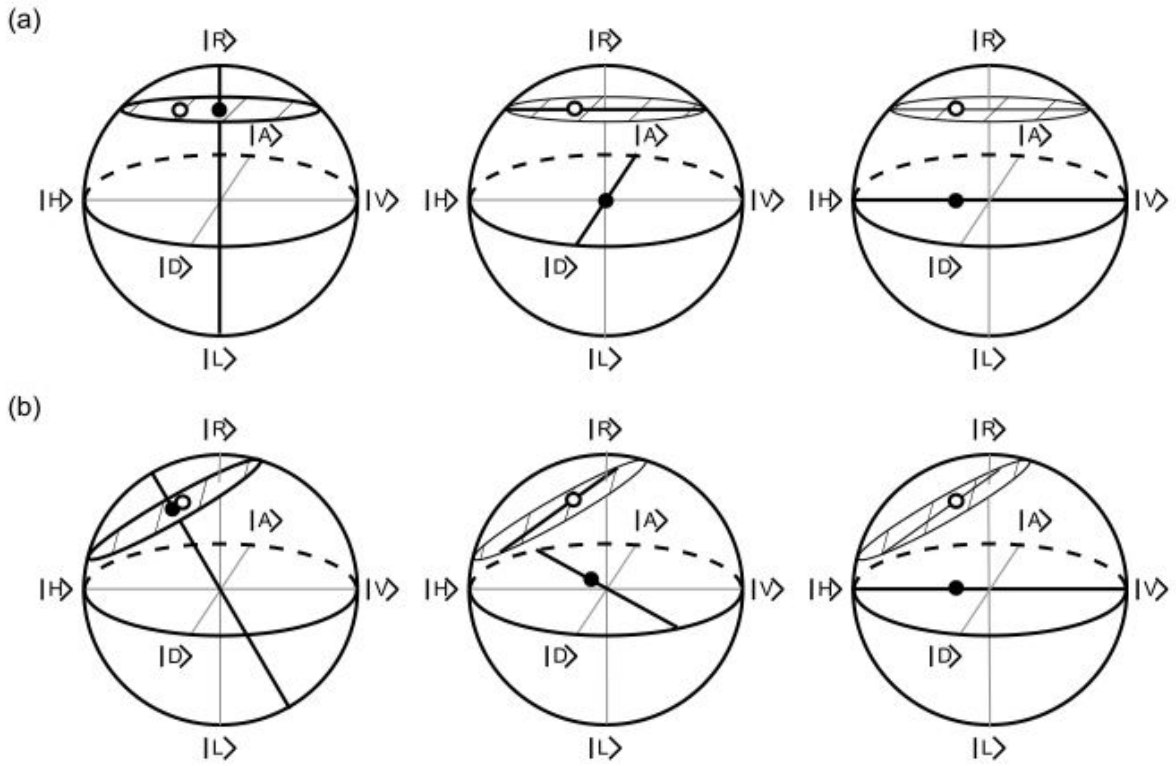
1.3.1 Visualisation of single qubit tomography

Exact single-qubit tomography requires a sequence of three linearly independent measurements. Each measurement exactly specifies one degree of freedom for the measured state, reducing the free parameters of the unknown

state's possible Hilbert space by one. As an example, consider measuring R, D, and H on the mixed state

$$\hat{\rho} = \begin{pmatrix} 5/8 & -i/\sqrt{2} \\ i/\sqrt{2} & 3/8 \end{pmatrix}$$

Then the Poincare sphere representation for this is:



This figure shows a sequence of three linearly

independent measurements isolate a single quantum state in Hilbert space (shown here as an open circle in the Poincare sphere representation). The first measurement isolates the unknown state to a plane perpendicular to the measurement basis. Further measurements isolate the state to the intersections of non-parallel planes, which for the second and third measurements correspond to a line and finally a point. The black dots shown correspond to the projection of the unknown state onto the measurement axes, which determines the position of the aforementioned planes. (a) A sequence of measurements along the right-circular, diagonal, and horizontal axes. (b) A sequence of measurements on the

same state taken using non-orthogonal projections: elliptical light rotated 30° from H towards R, 22.5° linear, and horizontal.

1.4 Exact Tomography of Multiple Qubits using

While the scheme outlined above is most efficient in the sense that it requires only 3^n analysis settings for the experimental apparatus, in practice it may only be experimentally possible to make a single projective measurement at a time. This will require $4^n - 1$ probabilities in order to define a complete set of T_i parameters. In practice, this will mean that 4^n measurements are necessary in order to normalize counts to probabilities. By making a set of single projective measurements on each qubit and only taking into account those results where a definite result is obtained (e.g., the photon was transmitted by the polarizer), it is possible to reconstruct a state using only n detectors.

1.5 Errors in real Tomography

The tools developed so far allow the perfect reconstruction of a density matrix from an infinite set of ideal data. When applying this technique to any set of real measurements, the assumption of ideal probabilities must be discarded. In fact, the probabilities predicted by real results can in practice be contradictory or even physically impossible. It is therefore necessary to implement a procedure that takes these errors into account yet always returns a legitimate density matrix.

1.6 Types of Errors

Errors in the measurement of a density matrix fall into three main categories: errors in the measurement basis, errors from counting statistics, and errors from experimental stability. The first problem can be addressed by increasing the accuracy of the measurement apparatus while the second problem is reduced by performing each measurement on a larger ensemble (counting for a longer time). The final difficulty is drift which occurs over the course of the tomography. This drift occurs either in the state produced or the efficiency of the detection system, and can constrain the data-collection time.

After all sources of error are taken into account, a single measurement results in a distribution over all possible states describing the experimenter's knowledge of the unknown state. This distribution represents the likelihood that a particular state would give the measured results, relative to another state. When independent measurements are combined, these distributions are multiplied, and ideally the knowledge of the unknown state is restricted to a small

ball in Poincare space, approximately equal to a three-dimensional Gaussian. This type of state isolation occurs regardless of which measurements are taken, as long as they are linearly independent. Even after tomography returns a distribution of likelihood over Poincare space, one final problem remains. It is very possible, especially with low counts or with the measurement of very pure states, that state estimation will return an illegal state. As all legal states have a radius of less than or equal to one in Poincare space, it is necessary to find a way to return the most likely legitimate state reconstructed from a set of measurements.

1.6.1 Maximum Likelihood Technique

The maximum likelihood technique finds the parameter of a probability distribution most likely to have given an experimental outcome. The problem of reconstructing illegal density of matrices is resolved by selecting the legitimate state most likely to have returned the measured counts. We require three elements: a manifestly legal parametrization of a density matrix, a likelihood function which can be maximized, and a technique for numerically finding this maximum over a search of the density matrix's parameters.

The probability of obtaining the observed experimental counts n_ν from the density matrix $\hat{\rho}$ is:

$$P(n_1, n_2, \dots, n_\Xi) = \frac{1}{Norm} \prod_{\nu} \exp \left[-\frac{(\bar{n}_\nu - n_\nu)^2}{2\hat{\sigma}_\nu^2} \right]$$

The likelihood that the matrix $\hat{\rho}_p(t_1, t_2, \dots, t_n)$ could produce the measured data $\{n_1, n_2, \dots, n_\Xi\}$ is:

$$P(n_1, n_2, \dots, n_\Xi) = \frac{1}{Norm} \prod_{\nu} \exp \left[-\frac{(\mathcal{N}\langle\psi_\nu|\hat{\rho}_p(t_1, t_2, \dots, t_n)|\psi_\nu\rangle - n_\nu)^2}{2\mathcal{N}\langle\psi_\nu|\hat{\rho}_p(t_1, t_2, \dots, t_n)|\psi_\nu\rangle} \right]$$

Rather than find the maximum value of $P(t_1, t_2, \dots, t_n)$, it is somewhat simpler to find the maximum of its logarithm. Thus, the optimization problem reduces to finding the minimum of the following function:

$$\mathcal{L}(t'_1, t'_2, \dots, t'_{n^2}) = \sum_{\nu} \frac{(\langle\psi_\nu|\hat{\rho}_p(t'_1, t'_2, \dots, t'_{n^2})|\psi_\nu\rangle - n_\nu)^2}{2\langle\psi_\nu|\hat{\rho}_p(t'_1, t'_2, \dots, t'_{n^2})|\psi_\nu\rangle}.$$

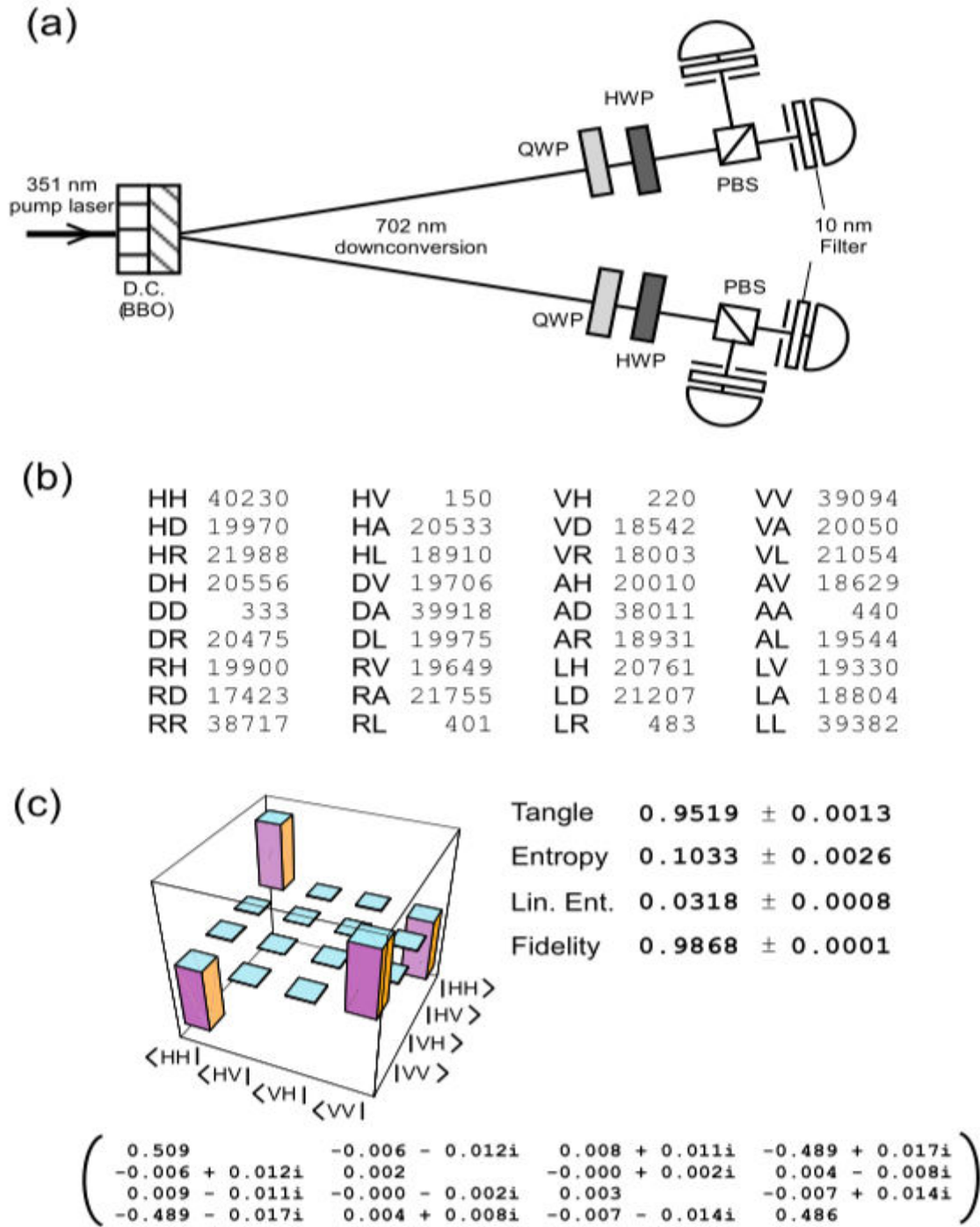
The final piece in the maximum likelihood technique is an optimization routine, of which there are many available.

1.7 Error analysis

Error analysis of reconstructed density matrices is in practice a non-trivial process. The traditional method of error analysis involves analytically solving for the error in each measurement due to each source of error, then propagating these errors through a calculation of any derived quantity. In practice, however, these errors appear to be too large: We have experimentally repeated some of our measurements many times, and observed a spread in the value of derived quantities which is approximately an order of magnitude smaller than the spread predicted from an analytic calculation of the uncertainty. Thus, it is worthwhile to discuss alternate methods of error analysis.

In this figure (a) Experimental setup for producing two-qubit polarization states from spontaneous parametric down conversion. (b) Experimental data for a near-Bell state. Shown are the single photon counts from a complete tomography using four detectors. Nine analysis settings yield the 36 measurement results shown. These counts are adjusted for accidentals (dark counts) and renormalized for differences in detector efficiencies (see text for details). (c) The density matrix is shown in both numeric and graphical form, along with several quantities derived from that matrix. The errors were calculated using a Monte Carlo simulation of the data from (b).

One promising numerical method is the ‘Monte Carlo’ technique, whereby additional numerically simulated data is used to provide a statistical distribution over any derived quantity. Once an error distribution is understood over a single measurement (e.g., Gaussian for waveplate setting errors or poisoning over count statistics), a set of ‘simulated’ results can be generated. These results are simulated using the known error distributions in such a way as to produce a full set of numerically generated data which could feasibly have come from the same system.



Many of these sets of data are numerically generated (at the measured counts level), and each set is used to calculate a density matrix via the maximum likelihood technique. This set of density matrices is used to calculate the standard error on any quantity implicit in or derived from the density matrix. Clearly, the problem of error analysis in state tomography is an area of continuing research – one of many. The development of adaptive tomography techniques could allow both specific measurements and the data collection times to be tailored in order to optimize for each state to be measured [37]. In addition, because the number of measurements necessary to perform tomography grows exponentially with the

number of qubits, it will eventually be necessary to partially characterize states with fewer measurements. Finally, each distinct qubit implementation provides a myriad of unique challenges. Nevertheless, we hope the discussions presented here will be useful for characterizing quantum systems in a broad spectrum of qubit realizations.

2. Randomized Benchmarking

2.1 Introduction

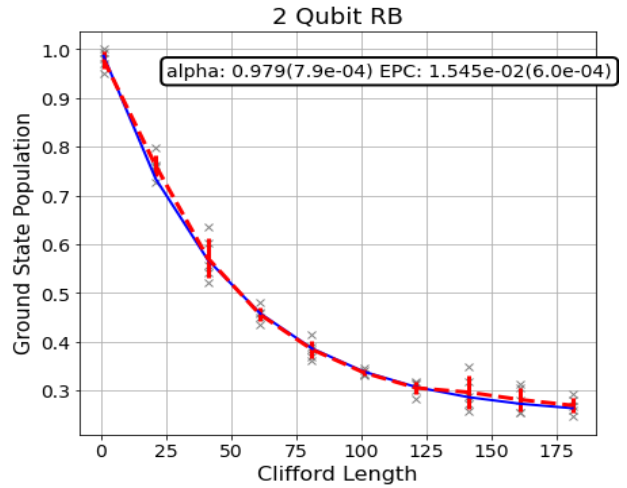
A key requirement for scalable quantum computing is that elementary quantum gates can be implemented with sufficiently low error. One method for determining the error behaviour of a gate implementation is to perform process tomography. However, standard process tomography is limited by errors in state preparation, measurement and one-qubit gates. It suffers from inefficient scaling with number of qubits and does not detect adverse error compounding when gates are composed in long sequences. An additional problem is due to the fact that desirable error probabilities for scalable quantum computing are of the order of 0.0001 or lower. Experimentally proving such low errors is challenging. We describe a randomized benchmarking method that yields estimates of the computationally relevant errors without relying on accurate state preparation and measurement. Since it involves long sequences of randomly chosen gates, it also verifies that error behaviour is stable when used in long computations. We implemented randomized benchmarking on 2-qubit simultaneous with 1-qubit, establishing a **Error per Clifford (EPC) 1.320763e-02**. In this particular experiment, we have 3 qubits Q0,Q1,Q2. We are running 2Q RB (on qubits Q0,Q2) and 1Q RB (on qubit Q1) simultaneously, where there are twice as many 1Q Clifford gates. We define a noise model for the simulator. To simulate decay, we add depolarizing error probabilities to the CNOT and U gates. We execute these sequences, but with a noise model extended with T1/T2 thermal relaxation error, and fit the exponentially decaying curve. We count the number of **gates per Clifford**, and calculate the **two-qubit Clifford gate error**, using the predicted primitive gate errors from the coherence limit.

2.2 What is Randomized Benchmarking?

A proven protocol that provides an efficient and reliable estimate of an average error-rate for a set of quantum gate operations.

Consists of the following three stages:

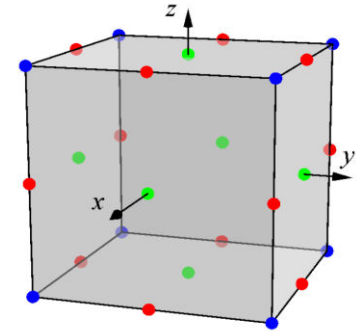
- Generate RB sequences consisting of random elements from a Clifford group, including a computed reversal gate
- Run the RB sequences either on the device or on a simulator (with a noise model) and compare to the initial state
- Get the statistics and fit an exponential decaying curve: Ap^m+B Error per Clifford (EPC): $r=(1-p)(d-1)/d$ ($d=2n$)



2.3 Clifford Group

- The **Clifford group** consists of the quantum operators that can be efficiently simulated (in polynomial time) using a classical computer –Clifford simulation.
- It is generated by the gates: H , S and $CNOT$:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad CNOT = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



- The Clifford group on 1-qubit has 24 elements (the rotational symmetries of the cube)
- The Clifford group on n-qubit is of size: $2^{n^2+2n} \prod_{j=1}^n (4^j - 1)$
- Efficient algorithms to generate and synthesize Clifford elements
- The Clifford group forms a *unitary 2-design–twirling* the finite Clifford group is equivalent to *twirling* the infinite unitary group

2.4 RB methods for estimating noise parameters

- **Interleaved RB** Estimating the average error of individual quantum gates



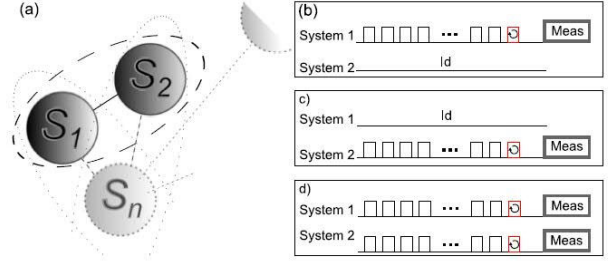
- **Purity RB** : Quantifies how coherent the errors are, namely

$$r_C^{\text{est}} = \frac{(d-1)(1 - p_{\overline{C}}/p)}{d},$$

$$\text{Tr}(\rho^2) = 1 \leftrightarrow \rho \text{ is pure (coherent)}$$

- **Leakage RB** : Quantification and characterization of leakage errors (unwanted energy levels)

- **Simultaneous RB** : Running RB simultaneously on subsets of qubits, characterizing the amount of addressability between subsystems



- **Correlated RB** : Estimating correlated crosstalk error between qubits participating in separate gates

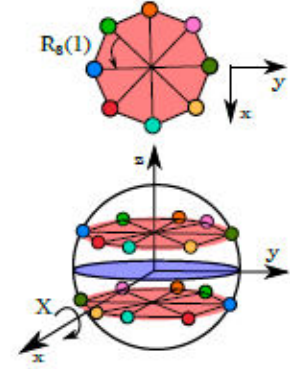
2.4 RB methods for other groups and gates

- **Non-Clifford Dihedral and CNOT-Dihedral RB**

$$D_m = \langle X, Z_m \rangle$$

$$Z_m = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/m} \end{bmatrix}$$

$$G_m = \langle CX(i, j), X(j), Z_m(j) \rangle$$

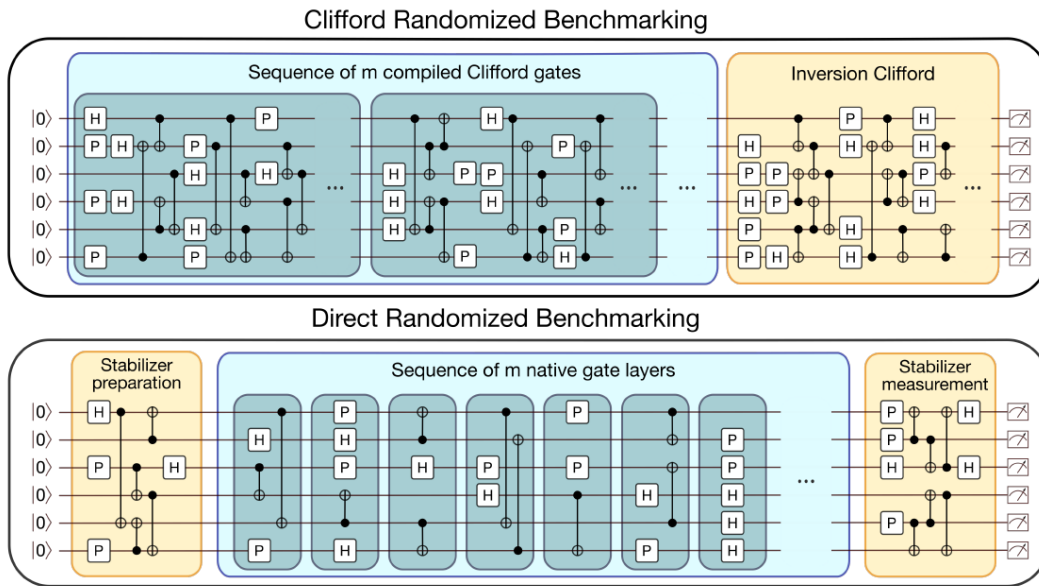


- Allows benchmarking of non-Clifford gates: T, CS, CCZ
- Efficient synthesis of CNOT-Dihedral elements
- **Pauli and CNOT-Pauli RB**: The Pauli group on n-qubits is generated by the tensor product of the Pauli matrices

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

- Subgroups of the Clifford group

- More efficient synthesis compared to the Clifford group



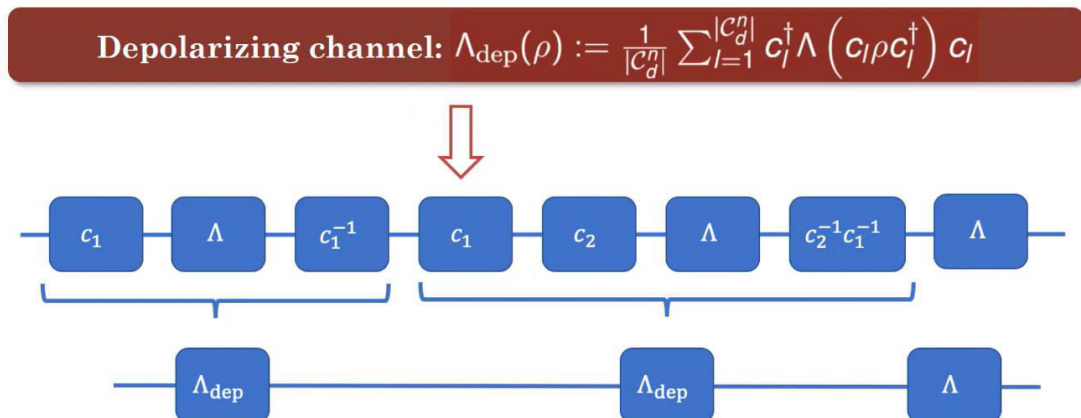
- **Direct RB:** Directly benchmarking the native gates of a device

2.5 Theoretical Background of RB

- **Ideal Clifford-gate Sequence**



- **Noisy Clifford-gate sequence for Noisy channel Λ**
- **By independence, this factorizes into product of average channels ('Twirl')**



- **Sequence of Twirled noisy channels:**
- **Depolarizing channel (Shur's Lemma):**

$$\Lambda(\rho) = p\rho + (1 - p)/d * I$$

- **The average data looks like a product of depolarizing channels:**

$$p^m \rho + (1 - p^m)/d * I$$



$$\Lambda \circ \Lambda_{\text{dep}}^{\circ m-1}(\rho) = p^{m-1} \Lambda(\rho) + (1 - p^{m-1}) \Lambda(I)/d^n$$

$$p^m(d - 1)/d + \frac{1}{d}$$

- **Fit the results:**

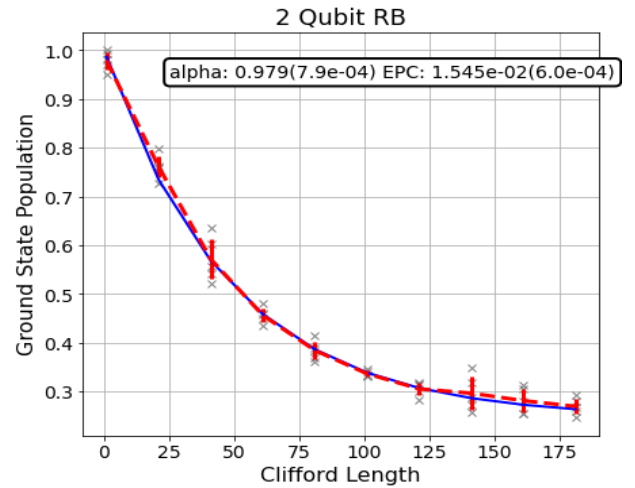
$$Ap^m + B$$

- **Calculate the average Error per Clifford (EPC):**

$$\begin{aligned} r &= (1 - p) - (1 - p)/d \\ &= (1 - p)(d - 1)/d \end{aligned}$$

- **Calculate the 2-qubit gate fidelity:**

$$\begin{aligned} r_{cx} &\approx r/n_{cx} \\ n_{cx} &\approx 1.5 \end{aligned}$$



2.6 Theoretical Background of the Clifford group

- The **Clifford group** C_n on n -qubits is defined as the normalizer of the **Pauli group** P_n , when neglecting the global phase $U(1) = \mathbb{C}$

$$P_n / U(1) = F_2^{2n}$$

$$C_n/P_n = Sp(2n)$$

- $Sp(2n)$ denotes the group $2n \times 2n$ **Symplectic** matrices over the field \mathbf{F}_2 , which is **simple** when $n > 2$.

- **1-qubit Clifford group (24 elements):**

Since $C_1/P_1 = Sp(2) = S_3$, we can explicitly present each elements of C_1 as a product AB where:

$$\begin{aligned} A \text{ in } \{I, V, W, H, HV, HW\} (V = S^\dagger H, W = HS) \quad , \\ B \text{ in } P_1 = \{I, X, Y, Z\} \end{aligned}$$

- **n-qubit Clifford group:**

Algorithms to efficiently select random elements of C_n of size:

$$2^{n^2+2n} \prod_{j=1}^n (4^j - 1)$$

Algorithms to synthesize elements of C_n (optimized for $n=1,2,3$).

3. Quantum Volume

3.1 Introduction

In light of recent development in Quantum Computers, it has become inevitable to think about a method with which one can compare Different type of QCs. This Method should be able to compare QCs no matter how different they are in terms of technologies involved.

Benchmarking of quantum Computer is not just Simply Comparing Qubits, therefor it becomes essential to have a method which measures and understands systematically all the factors affecting power and performance of a Quantum Computer. Quantum Volume is one such method which takes into account all relevant hardware parameters. This includes the *performance parameters* such as *coherence*, *calibration errors*, *crosstalk*, *spectator errors*, *gate fidelity*, *measurement fidelity*, *initialization fidelity* as well as the *design parameters* such as *connectivity* and *gate set*. It also includes the software behind the circuit optimization. Additionally, the quantum volume is architecture-independent, and can be applied to any system that is capable of running quantum circuits.

3.2 What is it and its importance

Quantum Volume is a numerical value which shows complexity of a problem. It basically gives maximum number of square quantum circuits that can be implemented successfully by the computer. Conventionally the benchmarking is done with number of qubits in the system. More qubits are required to solve increasingly difficult problems, and thus, everything else being equal, the more qubits a quantum computer has, the more computational power it has. But as the number increases the errors also increases and this has to be considered for benchmarking.

Calculation of Quantum volume is based upon two parameters, Width and depth. The Width is nothing but the number of qubits and Depth is measurement of length of quantum circuit executed for each qubit. Technically its the number of layers of the circuit. Measurement of quantum volume can be performed via

Quantum volume protocol. The Matric Quantum Volume is influenced by Connections between qubits which further depends upon types of Qubits used. Like if the architecture of qubits is of trapped-ion type then it can support any-to-any connectivity (flexibility to entangle any two qubits with a single gate even if they are not physically adjacent). While Superconducting qubits only provides nearest-neighbour connectivity which results in noise. On one Hand, qubits connected on a line would need significant overhead for any random pair of Qubit. While on the other hand if all the qubits are connected to each other then it would greatly influence metrics such as crosstalk, fidelity which may cause error in computation.

The Quantum Volume protocol is deeply connected to Gate Error rates. It is always an ideal situation to have lower error rates. One of the critical errors is Spectator error which can arise from qubits not participating in applied quantum gates. Hence Placement of Qubits also becomes important to reduce errors. To further reduce errors choice and performance of used gate sets is one of the very important aspect. A large set of gates reduces the overhead to move quantum information, but also requires far more complexity from a calibration. The internal states of a quantum computer are fragile and susceptible to noise, introducing errors that lead to incorrect answers. This is why Compilers are critical for optimal translation of circuits to the underlying hardware. A compiler needs to consider and optimize over the device connectivity, and potentially even variations of gate fidelities and spectator errors across the device. Quantum computing systems with high-fidelity operations, high connectivity, large calibrated gate sets, and circuit rewriting toolchains are expected to have higher quantum volumes.

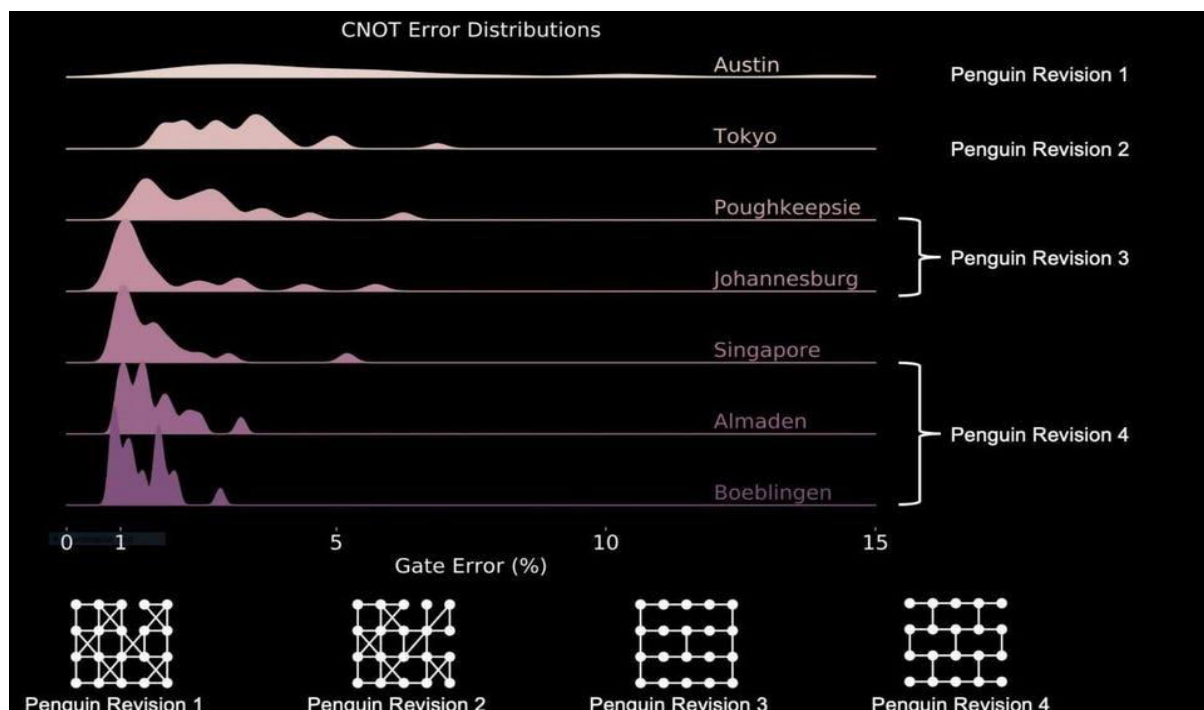
3.3 Calculation of Quantum Volume

Quantum Volume provides a way to compare devices across different physical implementations, and measures qualities, such as low error rates, that are ultimately necessary for a practical quantum computer. The Quantum Volume measures the largest model circuits the quantum computer can successfully run. A model circuit consists of random two-qubit gates acting on random pairs of qubits, and has as many parallelized layers of these gates as it has qubits. The model circuits are compiled to the particular quantum system. There is nothing special about the results derived from executing randomly-generated circuits, just that the results should be the same when executed on a real quantum computer as when executed on a quantum simulator. Not exactly the same, both because quantum computers are probabilistic by nature, and some degree of errors are to

be expected. A given run is considered successful if the observed measurement outcome is in the upper half of the ideal output probability distribution. To claim that the quantum volume exceeds some value for some system, we are required to succeed for more than two-thirds of the runs on a given number of qubits. In Simple term if the circuit is able to successfully simulate circuit with n qubits and n gate depth then quantum volume for such system is 2^n . So, 4×4 circuit would have QV of 16, a 5×5 circuit would have QV of 32.

3.4 Application of Quantum Volume

Quantum Volume can play a significant role in ongoing development and research necessary to create bigger and better quantum computers to achieve a quantum advantage. Figure 1 shows progressive improvement in CNOT error



rates as a result of various changes in qubit connectivity and improvement in Quantum Volume.

Fig. IBM Quantum

There are many other performance metrics and architectural choices that impact the quantum volume of a system as well - for instance how the qubits are laid out and connected to one another. This also helps to compare and understand how errors rates might change in different architectures. In Trapped ions, for instance will have low error rates because of having all-to-all connectivity. Which is not the case for Superconducting Architectures as shown in Figure below.

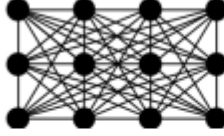
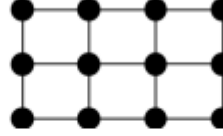
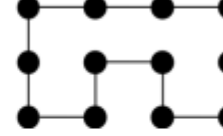
$\log_2 V_Q$	All-to-All	Square Grid	Loop
			
4	0.03	0.028	0.028
6	0.015	0.011	0.011
8	0.008	0.005	0.0047
12	0.0032	0.0015	0.0014

Fig. IBM Quantum

3.5 Limitation of Quantum Volume

Although Quantum Volume seems to be very useful metric for benchmarking Quantum Devices it does have some flaws. The Calculation of Quantum Volume metric is based upon Squared-Circuit Configuration but, in practice very few Quantum Algorithms have Square-Circuit configuration. Most of the algorithms either use a large number of Qubits but very few levels of gate depth or have a much larger number of gate operation but use very few qubits. Some of such algorithms are VQE, QAQA or Shor's Algorithm.

Another problem is that quantum simulators are incapable of simulating circuits using more than around 50 qubits since the number of quantum states grows exponentially as the number of qubits grows. To simulate 50 qubits the simulator needs to store one quadrillion state (one million billion), which is a lot of quantum states for a simulator to keep track of.

Hence, quantum volume is a plausible metric for smaller quantum computers and smaller quantum circuits, but clearly it won't work well and won't work at all for larger quantum computers and larger quantum algorithms.

3.6 Summary

The Quantum Volume provides a way to benchmark the whole quantum computing system, including the optimizing components of the software stack.

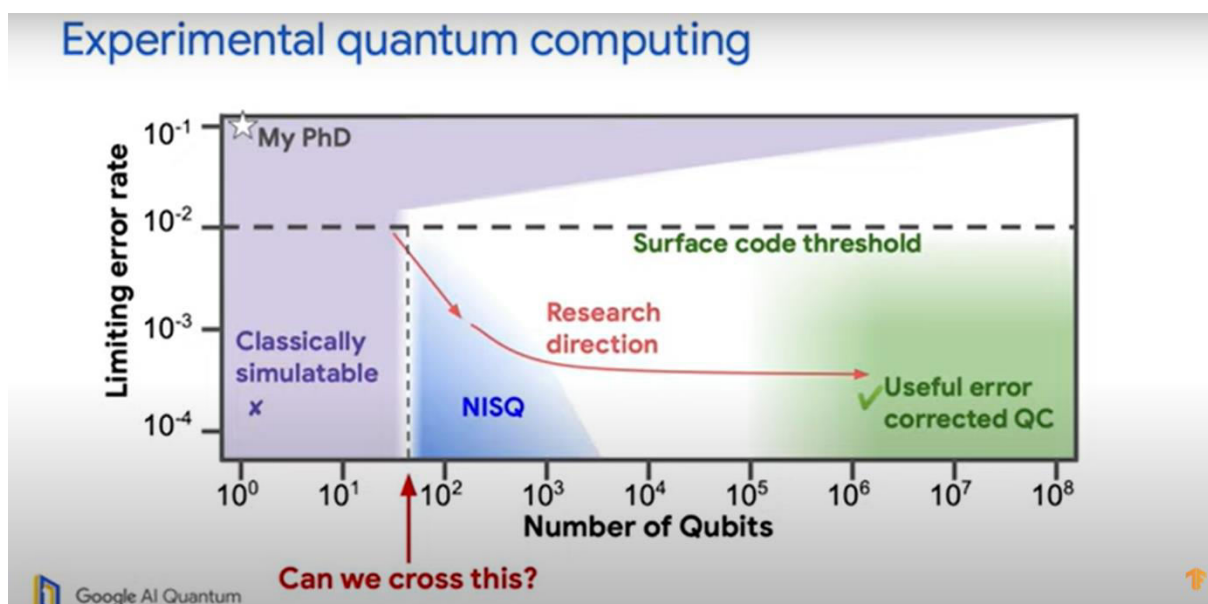
Choices for universal benchmarks will evolve as the community continues to learn more about near-term devices, but the quantum volume is an accessible and measurable quantity that tracks progress on current devices.

But, due to some shortcoming of Quantum volume metric, it is possible that it may be superseded by new volume metrics or it could be modified to overcome such shortcomings. Nonetheless a universal benchmarking scheme must be developed.

4. CROSS ENTROPY BENCHMARKING

4.1 Introduction

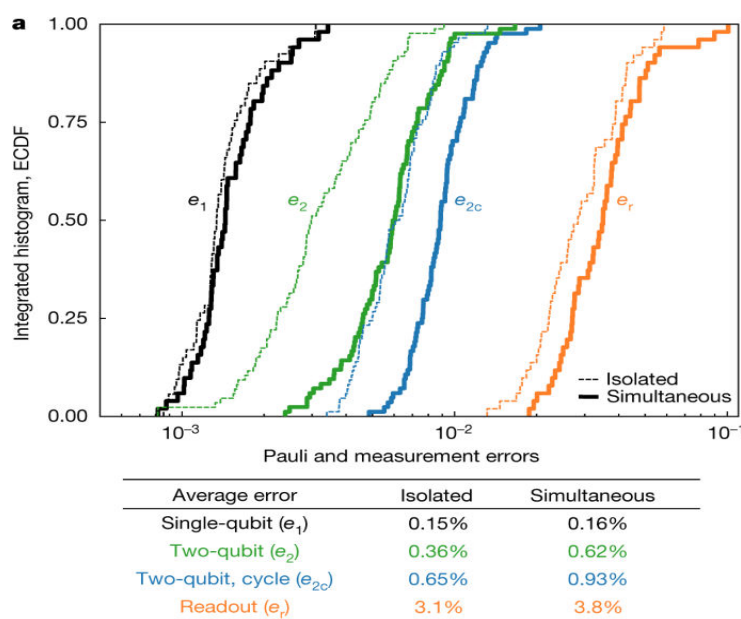
The purpose of quantum computing is to solve specific problems that are beyond the reach of classical computing. Most of these applications are in the **Useful error-corrected machine**, which is still years ahead in research. We also hope to find practical applications in NISQ (Noisy Intermediate Scale Quantum) area. The intermediate scale here indicates devices that are large enough, perhaps 50 qubits or more, that we can't simulate the quantum system using our most powerful existing digital computers. Our aim is that a quantum computer does anything stubborn or intractable for a classical computer. But the question arises if we could surpass the boundary between classically simulatable and beyond classically simulatable?



Credits: Google AI Quantum

This was named Quantum Supremacy by John Preskill in 2012. “*Quantum Supremacy is to perform tasks with controlled quantum systems going beyond what can be achieved with ordinary digital computers.*” He also asked if controlling large-scale quantum systems is merely really very hard or if it is ridiculously hard. A paper named ‘Quantum supremacy using a programmable superconducting processor’ demonstrated it on Sycamore processor and said that it is indeed merely really hard.

Benchmarking results: Let us start with single qubit gates. We plot the distributions of errors across the device for 53 qubits (Sycamore). For single qubit gate, there is an error of about 0.15% and if for simultaneous, it increases to 0.16%. This indicates that there are no interactions between the two qubits. For two qubit gates in the isolated case, there is an error of 0.36% while in simultaneous case, it increases to 0.62%. This increase in error is due to the interactions between pairs of qubits.



Credits: Wikipedia

4.2 Cross entropy benchmarking

This was basically looking at pairs one at a time and not having entanglement across the device. What we actually want to do is to evaluate the performance of the full processor doing a realistic algorithm. The method of cross entropy benchmarking is intractable. This is because of the fact that we have 50+

qubits and we are going to deal with 2^{53} is about 10^{16} different amplitudes or probabilities. It is almost impossible to resolve all these probabilities and so we sample from expected distribution. We get samples, bit strings and we compare them to the probabilities that we expect based on the experiment.

In cross entropy benchmarking (XEB), a random quantum circuit is executed on a quantum computer multiple times; like about a million times in order to collect a set of k samples in the form of $\{x_1, \dots, x_k\}$. The bitstrings observed are then used to estimate the fidelity using linear cross entropy.

$$\text{Fidelity} = 2^n \langle q_k \rangle_{\text{observations}} - 1$$

This fidelity is basically how often we sample the high probability bit strings. XEB can characterize the performance of a large device. When applied to deep, two-qubit circuits it can be used to accurately characterize a two-qubit interaction, thus leading to better calibration.

Sycamore processor was the first to demonstrate quantum supremacy via XEB. With $n=53$ and 20 cycles, it was able to run to obtain an XEB 0.0024. Generating samples took 200 seconds on the quantum processor when it would have taken 10,000 years on Summit; the classical supercomputer, at the time of the experiment. Improvements in classical algorithms have shortened the runtime to about a week on Sunway TaihuLight; collapsing Sycamore's claim to quantum supremacy.

In 2021, the latest demonstration of quantum supremacy by Zuchongzhi 2.1 $n=60$, 24 cycles and an XEB of 0.000366 holds. It takes around 4 hours to generate samples on Zuchongzhi 2 and 10,000 years on Sunway.

4.3 Clifford XEB

Linear cross-entropy benchmarking (XEB) is used extensively for systems with 50 or more qubits but is limited due to the exponentially large computational resources required for classical simulation. Recently, Chen and his teammates proposed a conducting linear XEB with Clifford circuits, called as Clifford XEB. Since Clifford circuits can be easily simulated classically, Clifford XEB can be scaled to much larger systems.

With this number of qubits, challenges arise like large number of control lines and pulse generators required to perform measurements, processor design, interactions between qubits and poor two-qubit gate connectivity. Considering these challenges, there is a demand for benchmarking assuring accurate results

when running algorithms. Such benchmarks can also provide information about whether there are correlated errors in the system by comparing the results to the digital error model. This is crucial for realizing fault-tolerant quantum computation.

We implemented linear XEB by running quantum circuits with random layers of gates and performed a computational basis measurement, then computing the probabilities of observed bitstrings. This is then repeated and an overall average is taken. This average is the linear XEB measure. It has been experimentally and numerically observed that this measure exponentially decays with the number of cycles for a noisy circuit, and this decay exponent is used as a measure of gate quality. Linear XEB has the advantage on shallow circuits, which is easy to implement on current processors.

For testing the scalability of Clifford XEB, numerical simulations of certain random Clifford circuits are performed on various topologies with depolarizing noise. We simulate circuits with up to 225 qubits. For noise levels comparable to current hardware, we can see clear exponential decays whose rates are consistent with the digital error model. We also conduct simulations on the 2-D grid topology with 1,225 qubits to further test the scalability of Clifford XEB. With promising numerical results, it is able to prove that Clifford XEB yields an exponential decay under a general error model for sufficiently low error. Clifford XEB is both efficient in quantum circuit size and necessary classical processing, and it has great promise for benchmarking quantum devices much larger than possible.

This is relevant as the number of qubits on processors continues to increase and the calibration process becomes more complex. Also, there are a variety of hurdles to overcome to be able to scale quantum processors to achieve fault-tolerance, limited size of dilution fridges and many more. Lastly, using a multiqubit benchmark helps verify that the quantum processor does not have large correlated errors which could harm quantum error correction.

5. Algorithms Based Quantum Computer Benchmarking

5.1 Introduction

Quantum computers experience a large number of errors at different levels. This can be because of miscalibration of hardware components like qubits or gates they act on. We can use benchmarking techniques like randomized benchmarking or gate set tomography for critical analysis of these low-level components. These techniques can provide insight into the type and magnitude of these errors. These tools are critical for experimental efforts to improve hardware.

But not all users are concerned about these low level details, rather most of the time user/researcher/investor are just concerned about how efficiently and successfully their application can be executed using quantum computer technology. Applying hardware level benchmarking techniques to predict the performance of a specific application is challenging, and often inaccurate. Also, not everyone can understand and interpret results of these hardware component level benchmarking as these are highly complex.

There has therefore been an increasing focus on benchmarks that concisely summarize the high-level performance of a quantum computer. A notable example is the quantum volume benchmark, which is designed to probe the effective useful size of a specific quantum computer and summarize it in one number: the quantum volume. The quantum volume has been widely adopted and reported by the community. However, due to the complexity of errors in quantum hardware, neither a device's quantum volume nor any other single metric is likely to accurately predict its performance on all applications.

There is thus a pressing need for a diverse set of application-oriented metrics and benchmarks that test the performance of quantum computers on

practically relevant tasks. Such benchmarks will enable hardware developers to quantify their progress in commercially relevant ways, and will make it possible for end users to more accurately predict how the available hardware will perform on their application. One way to approach this is to establish a set of quantum programs and measure the performance of a quantum computer when executing each one. This approach is Algorithm based quantum computer benchmarking. Here onward I will try to explain the work of Konstantinos Georgopoulos et al. [1] to give you an idea of how we can design algorithm based benchmarking frameworks.

5.2 Method

Though the work on these methods is not much in literature I will try to explain it using work by Konstantinos Georgopoulos et al. [1]. While approaching it the very first question arises is which quantum algorithms would be useful for program benchmarking quantum computers in the future as well. The choice of quantum algorithms is crucial, as we want them to be

1. **Scalability:** The algorithm should be able to scale up (or down) and run on increasingly larger quantum systems. scalable, in order to be able to face the challenge of the growing number of qubits in quantum computers
2. **Predictability:** The algorithm should produce a result that is easily predictable. An important addition to predictability is noise susceptibility: the algorithm should provide a result whose distortion under the effects of noise is easily distinguishable from the ideal evolution.
3. **Quantum Advantage:** The algorithm should provide a computational speed-up over its classical counterpart or, in other words, represent a possibly relevant real-world application.
4. On the basis of above mentioned properties, Konstantinos Georgopoulos et al. [1] choose five algorithms :
 1. Discrete-time Quantum walks
 2. Continuous-time Quantum Walks
 3. Pauli Decomposition of CTQW Hamiltonian.
 4. Quantum Phase Estimation
 5. Grover's algorithm

In their work they executed these five algorithms on two different IBM systems and tried to provide metrics to interpret and compare the system's efficiency in each case. The two IBM systems used in this are (a) IBMQ 5-qubit Bogota and Santiago machines and (b) the IBMQ 7-qubit Casablanca machine.

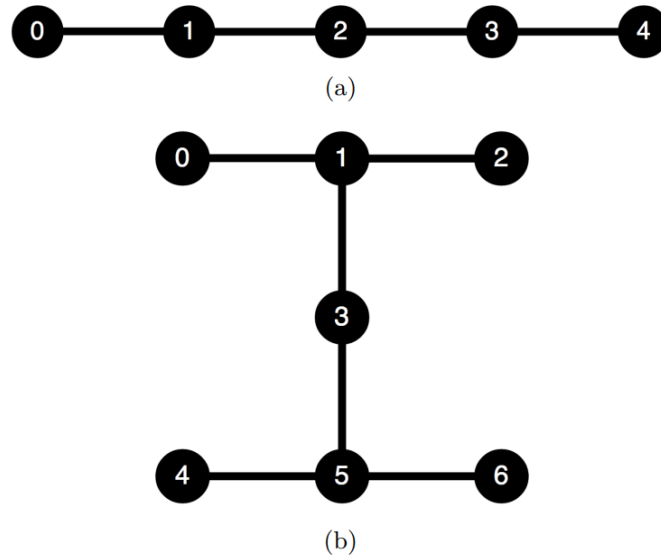


FIG. 1. Architectural graphs of the (a) IBMQ 5-qubit Bogota and Santiago machines and (b) the IBMQ 7-qubit Casablanca machine. A node in the graph represents a physical qubit whereas an edge represents a connection between a pair of qubits.

Source: IBM quantum

They identify four quantum circuit characteristics that are of interest for benchmarking:

1. The number of gates in the circuit
2. The number of active qubits, or qubits that are utilized by the quantum circuit (also called workspace)
3. The depth of the circuit, i.e., the longest path between the start of the circuit and a measurement gate
4. The runtime of the circuit on the quantum computer.

Machine	No. Gates	Size of Workspace	Depth	QC Runtime \pm s.d. (ms)
5-qubit Bogota	47	3	35	2.41 ± 0.03
5-qubit Santiago	47	3	35	2.34 ± 0.02
7-qubit Casablanca	47	3	35	2.54 ± 0.05

(a) Discrete-time quantum walk circuit characteristics.

Machine	No. Gates	Size of Workspace	Depth	QC Runtime \pm s.d. (ms)
5-qubit Bogota	19	2	13	2.34 ± 0.04
5-qubit Santiago	19	2	13	3.01 ± 0.07
7-qubit Casablanca	19	2	13	2.62 ± 0.01

(b) Continuous-time quantum walk circuit characteristics.

Machine	No. Gates	Size of Workspace	Depth	QC Runtime \pm s.d. (ms)
5-qubit Bogota	243	2	183	2.42 ± 0.04
5-qubit Santiago	243	2	183	2.28 ± 0.06
7-qubit Casablanca	243	2	183	3.76 ± 0.04

(c) Pauli decomposition of CTQW circuit characteristics.

Machine	No. Gates	Size of Workspace	Depth	QC Runtime \pm s.d. (ms)
5-qubit Bogota	93	4	66	2.54 ± 0.01
5-qubit Santiago	97	4	72	2.26 ± 0.03
7-qubit Casablanca	100	4	75	2.68 ± 0.06

(d) Quantum phase estimation circuit characteristics.

Machine	No. Gates	Size of Workspace	Depth	QC Runtime \pm s.d. (ms)
5-qubit Bogota	497	4	358	2.70 ± 0.08
5-qubit Santiago	479	4	343	2.56 ± 0.04
7-qubit Casablanca (ancilla)	788	6	503	2.81 ± 0.04
7-qubit Casablanca (no ancilla)	465	4	336	2.74 ± 0.03

(e) Quantum search circuit characteristics.

TABLE II. Quantum circuit characteristics for the five quantum circuits. No. gates and size of workspace are the number of gates and active qubits in the circuit respectively; depth of the circuit is the longest path between the start of the circuit and a measurement gate; QC runtime is the approximate average execution time of the circuit on the quantum computer along with standard deviation (s.d.).

5.3 Benchmark Indicators

The end results of the benchmarking process will be in the form of a comparison of the quantum computer output distribution with the distributions resulting from the unified noise model simulations of each machine and the ideal evolution. To compare the probability distributions and generate the benchmarks, we use the Hellinger distance (HD) [3].

Definition 1 (alpha benchmark)

The Hellinger distance between the probability distribution of the quantum computer evolution, Q , and the ideal distribution, D , namely $h_{id}(Q, D)$, is the alpha benchmark, with notation $\alpha_{q|i}$:

$$\alpha_{q|i} \equiv h(Q, D). \quad (4)$$

Definition 2 (beta benchmark)

The Hellinger distance between the probability distribution of the quantum computer evolution, Q , and the distribution resulting from the noisy simulations, N , namely $h_{nm}(Q, N)$, is the beta benchmark, with notation $\beta_{q|n}$:

$$\beta_{q|n} \equiv h(Q, N). \quad (5)$$

Definition 2 (gamma benchmark)

The Hellinger distance between the distribution resulting from the noisy simulations, N , and the ideal distribution, D , namely $h_{sm}(N, D)$, is the gamma benchmark, with notation $\gamma_{n|i}$:

$$\gamma_{n|i} \equiv h(N, D). \quad (6)$$

As mentioned in Section II E, the Hellinger distance is a metric that satisfies the triangle inequality. Hence, since the benchmarks established in the above definitions describe the pairwise Hellinger distances between three probability distributions (the quantum computer, Q , the simulated evolution, N , and the ideal evolution, D), it is possible to derive a relationship between $\alpha_{q|i}$, $\beta_{q|n}$ and $\gamma_{n|i}$ via the triangle inequality, through the following Lemma (whose proof follows simply from the triangle inequality for a metric).

5.4 Interpretation

Following the benchmarking framework we can extract meaningful results from a series of comparisons between the benchmark metrics, $\alpha_{q|i}$, $\beta_{q|n}$ and $\gamma_{n|i}$. The $\beta_{q|n}$ benchmark essentially describes how closely the noise model simulates the behavior of the quantum computer. The $\alpha_{q|i}$ benchmark shows how far the behavior of the quantum computer falls from the noise-free case thus giving an estimate of the overall computer performance under the effects of noise.

Also, we can interpret the triangle inequality as a measure of the confidence on the calibrated parameters encapsulating a picture of the noise that is accurate enough to provide a good estimation of the quantum evolution, expressed as follows:

- If $\beta_{q|n} \geq |\alpha_{q|i} - \gamma_{q|i}|$, then the over- or underestimation of the noise is small enough to provide estimates of the quantum evolution with high confidence.
- If $\beta_{q|n} < |\alpha_{q|i} - \gamma_{q|i}|$, then the calibrated parameters generate low confidence on the levels of noise

For the benchmarking experiments they run the quantum circuits that implement the chosen quantum algorithms. Results has been shared below :

Algorithm	No. Qubits	Size of Workspace	Iterations/Duration	Probability of Success
DTQW	2	3	1 (one coin-flip)	$p_{succ} = 0.5$ in states $ 1\rangle$ and $ 3\rangle$
CTQW	2	2	$t = 3$	$p_{ 2\rangle} = 0.99$, $p_{ 1\rangle} = p_{ 3\rangle} = 0.005$
PD	2	2	$t = 3$	$p_{ 2\rangle} = 0.99$, $p_{ 1\rangle} = p_{ 3\rangle} = 0.005$
QPE	3	4	1	$p_{succ} = 0.688$ in state $ 3\rangle$
QSa	4	6	3	$p_{succ} = 0.96$ in state $ 10\rangle$
QSn	4	4	3	$p_{succ} = 0.96$ in state $ 10\rangle$

TABLE III. The initial configuration for the quantum walk (DTQW), continuous-time quantum walk (CTQW) and its Pauli decomposition (PD), quantum phase estimation (QPE) and quantum search algorithms with ancilla (QSa) and without ancilla (QSn) for the benchmarking experiments. No. qubits is the number of qubits in the state space, size of workspace is the number of active qubits utilized by the circuit, iterations is the number of repetitions of the quantum circuit and the probability of success is the theoretical probability of the algorithm to give us the correct (or expected) result.

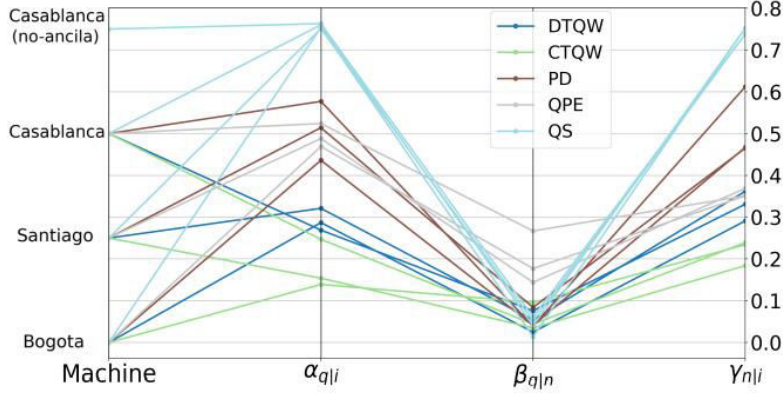


FIG. 2. Visualization of the three benchmarks, $\alpha_{q|i}$, $\beta_{q|n}$ and $\gamma_{n|i}$ for each of the three quantum computers (mapped to the left-hand side y -axis) when executing the five quantum algorithms.

Machine	$\alpha_{q i}$	$\beta_{q n}$	$\gamma_{n i}$	$ \alpha_{q i} - \gamma_{n i} $	QC Runtime (sec)	Sim. Runtime (sec)
5-qubit Bogota	0.287	0.025	0.291	0.004	240.7	3,331.5
5-qubit Santiago	0.321	0.054	0.362	0.041	234.2	3,119.3
7-qubit Casablanca	0.269	0.076	0.331	0.062	254.5	4,626.6

(a) Quantum walk algorithm on two qubits.

Machine	$\alpha_{q i}$	$\beta_{q n}$	$\gamma_{n i}$	$ \alpha_{q i} - \gamma_{n i} $	QC Runtime (sec)	Sim. Runtime (sec)
5-qubit Bogota	0.139	0.096	0.234	0.095	234.4	3,026.9
5-qubit Santiago	0.154	0.035	0.184	0.030	301.7	2,905.7
7-qubit Casablanca	0.247	0.044	0.239	0.008	262.1	3,035.2

(b) Continuous-time quantum walk algorithm on two qubits.

Machine	$\alpha_{q i}$	$\beta_{q n}$	$\gamma_{n i}$	$ \alpha_{q i} - \gamma_{n i} $	QC Runtime (sec)	Sim. Runtime (sec)
5-qubit Bogota	0.436	0.041	0.467	0.031	242.7	11,764.9
5-qubit Santiago	0.514	0.084	0.465	0.049	228.5	11,618.6
7-qubit Casablanca	0.577	0.039	0.612	0.035	376.4	12,861.3

(c) Pauli decomposition of the CTQW algorithm on two qubits.

Machine	$\alpha_{q i}$	$\beta_{q n}$	$\gamma_{n i}$	$ \alpha_{q i} - \gamma_{n i} $	QC Runtime (sec)	Sim. Runtime (sec)
5-qubit Bogota	0.469	0.177	0.351	0.118	254.2	6,287.1
5-qubit Santiago	0.488	0.144	0.369	0.119	226.8	6,304.2
7-qubit Casablanca	0.524	0.267	0.351	0.173	268.3	7,741.1

(d) Quantum phase estimation algorithm on three qubits.

Machine	$\alpha_{q i}$	$\beta_{q n}$	$\gamma_{n i}$	$ \alpha_{q i} - \gamma_{n i} $	QC Runtime (sec)	Sim. Runtime (sec)
5-qubit Bogota	0.754	0.014	0.752	0.002	270.4	22,834.3
5-qubit Santiago	0.751	0.051	0.738	0.013	256.7	21,923.2
7-qubit Casablanca (ancilla)	0.760	0.040	0.752	0.008	280.1	36,425.7
7-qubit Casablanca (no ancilla)	0.763	0.061	0.738	0.025	274.1	22,889.9

(e) Quantum search algorithm on four qubits.

TABLE IV. Results from benchmarking the three machines for each of the five algorithms. The benchmark indicators $\alpha_{q|i}$, $\beta_{q|n}$ and $\gamma_{n|i}$ are the HD between the quantum computer and the UNM, the quantum computer and the ideal distribution and the UNM and the ideal distribution respectively. QC runtime is the cumulative execution time in seconds for 100,000 iterations of each algorithm on the respective machine; Sim. runtime is the time it takes to simulate for 100,000 iterations the behaviour of each machine when executing the algorithms.

5.5 Summary

In their work they provide benchmarking metrics on the basis of which we can highlight efficiency of each system. Also we can compare execution of different algorithms on different circuits. But the question is how reliable these metrics are as quantum technology is changing very fast. Also how reliable are these metrics that is whether these five algorithms are sufficient to cover all applications of quantum computers. And the answer is as of now. Researchers were using different algorithms and trying to come up with a more standard and reliable framework. Also different researchers use different families of algorithms to cover a wider set of applications. So this framework is neither exhaustive in terms of covering application of Quantum computing nor does it provide any explanation of its reliability with advances in quantum technology in future. With this we can have an idea on how we can design an application based high level benchmarking framework using algorithms.

Conclusion

Measuring the progress of quantum computers can prove tricky in the era of “noisy” quantum computing technology because noise in the form of heat or electromagnetic sources constantly threatens to disrupt the computations among the fragile qubits, which makes it hard to reliably measure a quantum computer’s capabilities based only upon the total number of qubits.

Quantum Computers are using entirely different technologies to perform calculation hence, just one benchmarking technique will not do justice. Currently we have several methods which uses entirely different concepts to measure performance of any system which is not a very convenient and can lead to misinformation. Therefore, it becomes essential to have a universal benchmarking technique which can measure overall performance of any system having different architecture and technology involved and is easy to implement on every Systems.

Bibliography

- Arute, F., Arya, K., Babbush, R.; et al. (2019). "Quantum supremacy using a programmable superconducting processor". *Nature*. **574** (7779):505510. [arXiv:1910.11333](#). [Bibcode:2019Natur.574..505A](#). [doi:10.1038/s4158601916665](#). [PMID 31645734](#). [S2CID 204836822](#)
- Arnaud Carignan-Dugas, Joel Wallman, and Joseph Emerson, Characterizing Universal Gate Sets via Dihedral Benchmarking, *Phys. Rev. A* 92(6), 2015. <https://arxiv.org/abs/1508.06312>
- Boixo, S., Isakov, S.V., Smelyanskiy, V.N.; et al. (2018). "Characterizing Quantum Supremacy in Near-Term Devices". *Nature Phys.* **14** (6): 595–600. [arXiv:1608.00263](#). [Bibcode:2018NatPh..14..595B](#). [doi:10.1038/s4156018-0124-x](#). [S2CID 4167494](#).
- Challenges and Opportunities of Near-Term Quantum Computing Systems, [arXiv:1910.02894](#) [**quant-ph**]
- Easwar Magesan, Jay M. Gambetta, B. R. Johnson, Colm A. Ryan, Jerry M. Chow, Seth T. Merkel, Marcus P. da Silva, George A. Keefe, Mary B. Rothwell, Thomas A. Ohki, Mark B. Ketchen, and M. Steffen, Efficient measurement of quantum gate error by interleaved randomized benchmarking, *Phys. Rev. Lett.* 109(080505), 2012.
- <https://arxiv.org/abs/1203.4550>
- F. Arute, K. Arya, R. Babbush, and et. al., Quantum supremacy using a programmable superconducting processor, *Nature* 574, 505 (2019)
- G. White, D. F. V. James, P. H. Eberhard and P. G. Kwiat.
- Georgopoulos, K., Emary, C. and Zuliani, P., 2022. Quantum Computer Benchmarking via Quantum Algorithms. [online] [arXiv.org](#). Available at: <https://arxiv.org/abs/2112.09457>
- Jianxin Chen, dawei Ding, Cupjin Huang, Linghang Kong” Linear Cross Entropy Benchmarking with Clifford Circuits” <https://arxiv.org/abs/2206.082931>
- J. B. Altepeter, D. F. V. James and P. G. Kwiat.
- Jay M. Gambetta, A. D. C’orcoles, S. T. Merkel, B. R. Johnson, John A. Smolin, Jerry M. Chow, Colm A. Ryan, Chad Rigetti, S. Poletto, Thomas A. Ohki, Mark B. Ketchen, and M. Steffen, Characterization of addressability

by simultaneous randomized benchmarking, Phys. Rev. Lett. 109(240504) 2012, <https://arxiv.org/pdf/1204.6308>

- K. Wright, K. M. Beck, S. Debnath, and et. al., Benchmarking an 11-qubit quantum computer, Nature Communications 10, 5464 (2019).
- Lubinski, T., Johri, S., Varosy, P.D., Coleman, J., Zhao, L., Necaie, J., Baldwin, C.H., Mayer, K.H., & Proctor, T.J. (2021). Application-Oriented Performance Benchmarks for Quantum Computing.
- Liu X., Guo C. et al. (2021). "Redefining the Quantum Supremacy Baseline with a New Generation Sunway Supercomputer". [arXiv:2111.01066](https://arxiv.org/abs/2111.01066).
- M. Rötteler, Quantum algorithms for highly non-linear boolean functions, in Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (2010) pp. 448–457.
- Mahnaz Jaafarzadeh Randomized Benchmarking For Qudit Clifford Gates et al 2020 New J. Phys.. 22 063014 <https://iopscience.iop.org/article/10.1088/1367-2630/ab8ab1>
- Shelly Garion and Andrew W. Cross, *On the Structure of the Non-Clifford CNOT-Dihedral Group*, in preparation.
- Scott Aaronson and Daniel Gottesman, *Improved Simulation of Stabilizer Circuits*, Phys. Rev. A 70(052328), 2004. <https://arxiv.org/abs/quant-ph/0406196>
- Subdivided Phase Oracle for NISQ Search Algorithms, [arXiv:2001.06575](https://arxiv.org/abs/2001.06575) [quant-ph]
- The Pitfalls of Overreliance on the Quantum Volume Metric, Quantum Computing Report
- Timothy J. Proctor, Arnaud Carignan-Dugas, Kenneth Rudinger, Erik Nielsen, Robin Blume-Kohout, and Kevin Young, *Direct randomized benchmarking for multi-qubit devices*, Phys. Rev. Lett. 12(030503) 2019, <https://arxiv.org/abs/1807.07975>
- W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," Phys. Rev. A, vol. 100, no. 032328, 2019.
- The Pitfalls of Overreliance on the Quantum Volume Metric, Quantum Computing Report
- Z.-X. Jin and S.-M. Fei, Quantifying quantum coherence and nonclassical correlation based on Hellinger distance, Physical Review A 97, 10.1103/physreva.97.062342 (2018).

Appendix

RB Code for Simulations

#Import general libraries (needed for functions)

```
import numpy as np
import matplotlib.pyplot as plt
from IPython import display
#Import Qiskit classes
import qiskit
from qiskit.providers.aer.noise import NoiseModel
from qiskit.providers.aer.noise.errors.standard_errors import depolarizing_error, thermal_relaxation_error
```

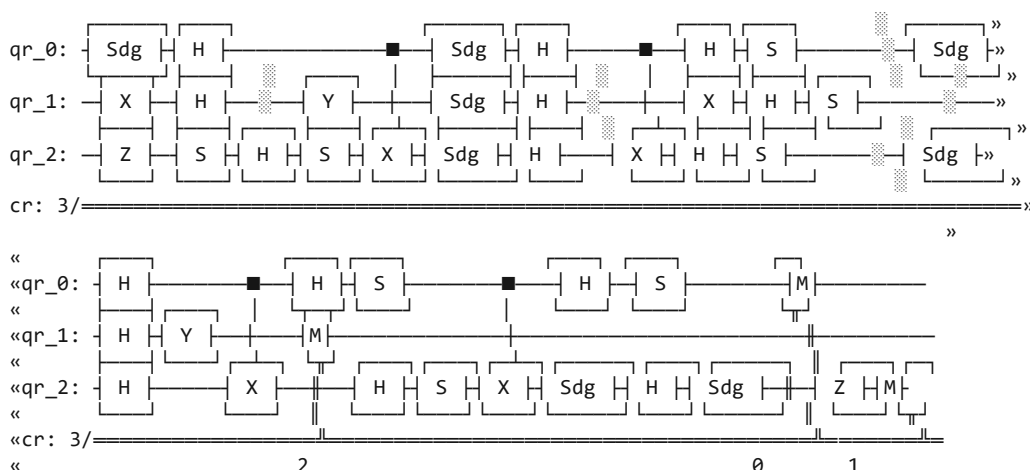
#Import the RB Functions

```
import qiskit.ignis.verification.randomized_benchmarking as rb
```

#Number of qubits

```
nQ = 3
#There are 3 qubits: Q0,Q1,Q2.
#Number of seeds (random sequences)
nseeds = 5
#Number of Cliffords in the sequence (start, stop, steps)
nCliffs = np.arange(1,200,20)
#2Q RB on Q0,Q2 and 1Q RB on Q1
rb_pattern = [[0,2],[1]]
#Do three times as many 1Q Cliffords
length_multiplier = [1,3]

rb_opts = {}
rb_opts['length_vector'] = nCliffs
rb_opts['nseeds'] = nseeds
rb_opts['rb_pattern'] = rb_pattern
rb_opts['length_multiplier'] = length_multiplier
rb_circs, xdata = rb.randomized_benchmarking_seq(**rb_opts)
print(rb_circs[0][0])
```



#Create a new circuit without the measurement

```
qc = qiskit.QuantumCircuit(*rb_circs[0][-1].qregs,*rb_circs[0][-1].cregs)
for i in rb_circs[0][-1][0:-nQ]:
```

```
qc.data.append(i)
```

```
#The Unitary is an identity (with a global phase)
```

```
backend = qiskit.Aer.get_backend('unitary_simulator')
basis_gates = ['u1', 'u2', 'u3', 'cx'] # use U,CX for now
job = qiskit.execute(qc, backend=backend, basis_gates=basis_gates)
print(np.around(job.result().get_unitary(), 3))

[[-1.+0.j -0.-0.j -0.-0.j -0.-0.j  0.+0.j  0.-0.j -0.-0.j  0.+0.j]
 [ 0.-0.j -1.+0.j -0.+0.j -0.-0.j -0.+0.j -0.+0.j -0.-0.j  0.+0.j]
 [ 0.-0.j -0.-0.j -1.+0.j -0.-0.j -0.-0.j -0.+0.j -0.+0.j -0.-0.j]
 [-0.-0.j  0.+0.j  0.-0.j -1.+0.j  0.-0.j -0.-0.j -0.+0.j  0.-0.j]
 [ 0.+0.j -0.+0.j -0.-0.j  0.+0.j -1.+0.j -0.+0.j -0.-0.j -0.-0.j]
 [ 0.-0.j -0.-0.j -0.+0.j -0.-0.j  0.-0.j -1.+0.j -0.+0.j -0.-0.j]
 [ 0.-0.j -0.+0.j -0.+0.j  0.+0.j  0.-0.j -0.-0.j -1.+0.j -0.-0.j]
 [-0.+0.j  0.+0.j  0.+0.j -0.-0.j -0.-0.j  0.+0.j  0.+0.j -1.+0.j]]
```

```
noise_model = NoiseModel()
p1Q = 0.002
p2Q = 0.01
noise_model.add_all_qubit_quantum_error(depolarizing_error(p1Q, 1), 'u2')
noise_model.add_all_qubit_quantum_error(depolarizing_error(2*p1Q, 1), 'u3')
noise_model.add_all_qubit_quantum_error(depolarizing_error(p2Q, 2), 'cx')
```

```
backend = qiskit.Aer.get_backend('qasm_simulator')
basis_gates = ['u1', 'u2', 'u3', 'cx'] # use U,CX for now
shots = 200
result_list = []
transpile_list = []
import time
for rb_seed, rb_circ_seed in enumerate(rb_circs):
    print('Compiling seed %d'%rb_seed)
    rb_circ_transpile = qiskit.transpile(rb_circ_seed, basis_gates=basis_gates)
    print('Simulating seed %d'%rb_seed)
    job = qiskit.execute(rb_circ_transpile, noise_model=noise_model, shots=shots, backend=backend,
backend_options={'max_parallel_experiments': 0})
    result_list.append(job.result())
    transpile_list.append(rb_circ_transpile)
print("Finished Simulating")
```

```
Compiling seed 0
Simulating seed 0
Compiling seed 1
Simulating seed 1
Compiling seed 2
Simulating seed 2
Compiling seed 3
Simulating seed 3
Compiling seed 4
Simulating seed 4
Finished Simulating
```

```
#Create an RBFitter object with 1 seed of data
```

```
rbfit = rb.fitters.RBFitter(result_list[0], xdata, rb_opts['rb_pattern'])
```

```
plt.figure(figsize=(15, 6))
```

```
for i in range(2):
    ax = plt.subplot(1, 2, i+1)
    pattern_ind = i
```

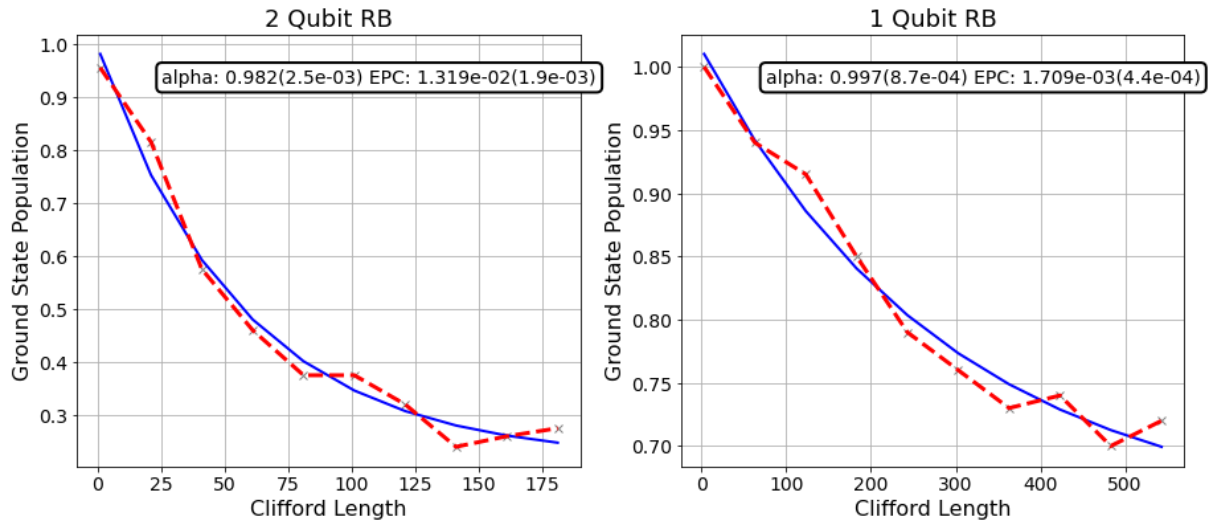
```
# Plot the essence by calling plot_rb_data
```

```
rbfit.plot_rb_data(pattern_ind, ax=ax, add_label=True, show_plt=False)
```

```
# Add title and label
```

```
ax.set_title('%d Qubit RB'%(len(rb_opts['rb_pattern'])[i])), fontsize=18)
```

```
plt.show()
```



```
rbfit = rb.fitters.RBFitter(result_list[0], xdata, rb_opts['rb_pattern'])
```

```
for seed_num, data in enumerate(result_list):#range(1,len(result_list)):
```

```
    plt.figure(figsize=(15, 6))
```

```
    axis = [plt.subplot(1, 2, 1), plt.subplot(1, 2, 2)]
```

```
    # Add another seed to the data
```

```
    rbfit.add_data([data])
```

```
    for i in range(2):
```

```
        pattern_ind = i
```

```
        # Plot the essence by calling plot_rb_data
```

```
        rbfit.plot_rb_data(pattern_ind, ax=axis[i], add_label=True, show_plt=False)
```

```
        # Add title and label
```

```
        axis[i].set_title('%d Qubit RB - after seed %d'%(len(rb_opts['rb_pattern'])[i]), seed_num), fontsize=18)
```

```
    # Display
```

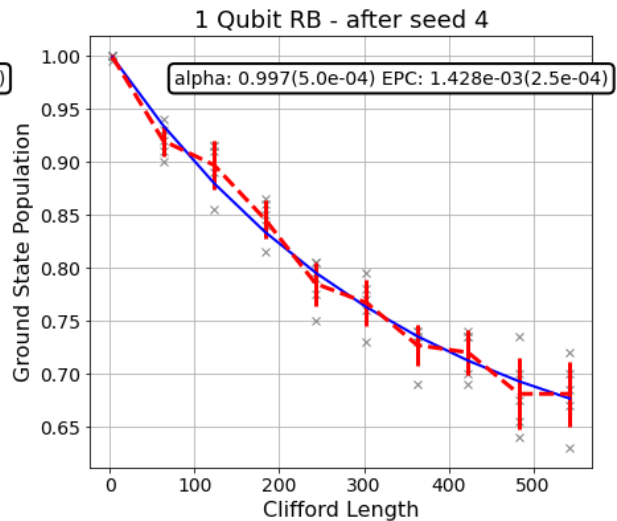
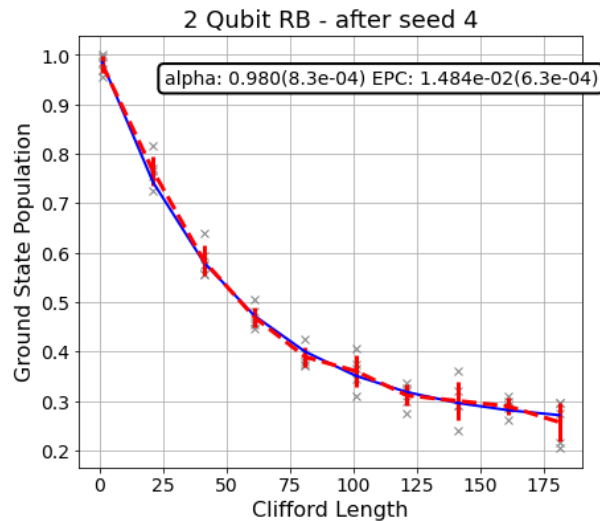
```
    display.display(plt.gcf())
```

```
    # Clear display after each seed and close
```

```
    display.clear_output(wait=True)
```

```
    time.sleep(1.0)
```

```
    plt.close()
```



```
shots = 200
result_list = []
transpile_list = []
for rb_seed, rb_circ_seed in enumerate(rb_circs):
    print('Compiling seed %d'%rb_seed)
    rb_circ_transpile = qiskit.transpile(rb_circ_seed, basis_gates=basis_gates)
    print('Simulating seed %d'%rb_seed)
    job = qiskit.execute(rb_circ_transpile, noise_model=noise_model, shots=shots, backend=backend,
backend_options={'max_parallel_experiments': 0})
    result_list.append(job.result())
    transpile_list.append(rb_circ_transpile)
print("Finished Simulating")
```

```
Compiling seed 0
Simulating seed 0
Compiling seed 1
Simulating seed 1
Compiling seed 2
Simulating seed 2
Compiling seed 3
Simulating seed 3
Compiling seed 4
Simulating seed 4
Finished Simulating
```

```
#Add this data to the previous fit
rbfit.add_data(result_list)
```

```
#Replot
plt.figure(figsize=(15, 6))
```

```
for i in range(2):
    ax = plt.subplot(1, 2, i+1)
    pattern_ind = i
```

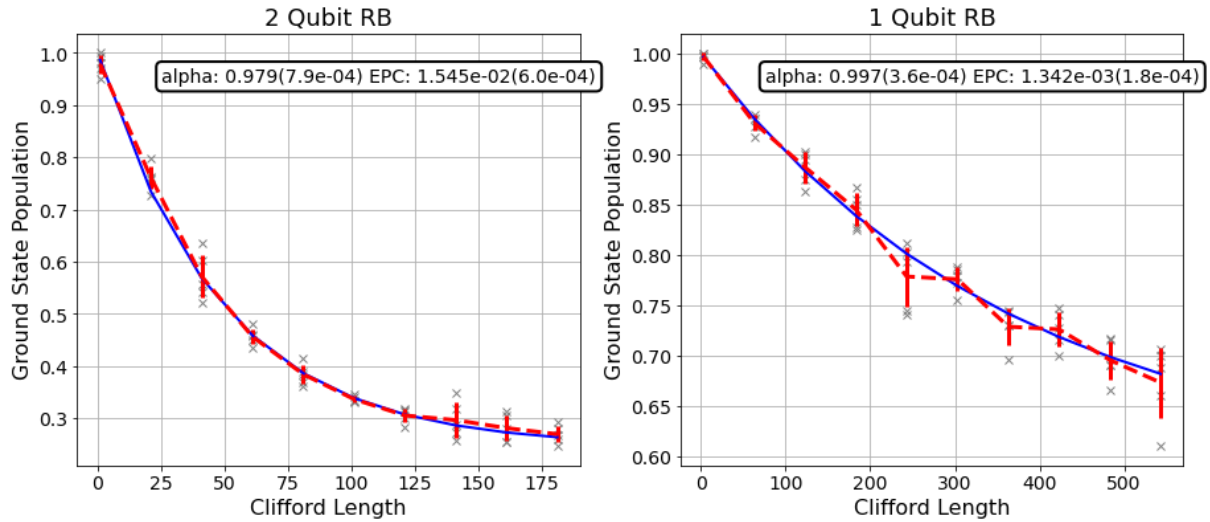
```
# Plot the essence by calling plot_rb_data
```

```
rbfit.plot_rb_data(pattern_ind, ax=ax, add_label=True, show_plt=False)
```

```
# Add title and label
```

```
ax.set_title('%d Qubit RB'%(len(rb_opts['rb_pattern'][i])), fontsize=18)
```

```
plt.show()
```



#Count the number of single and 2Q gates in the 2Q Cliffords

```
gates_per_cliff = rb.rb_utils.gates_per_clifford(transpile_list,xdata[0],basis_gates,rb_opts['rb_pattern'][0])
for basis_gate in basis_gates:
    print("Number of %s gates per Clifford: %f"%(basis_gate ,
        np.mean([gates_per_cliff[0][basis_gate],
            gates_per_cliff[2][basis_gate]])))
```

Number of u1 gates per Clifford: 0.142174
 Number of u2 gates per Clifford: 1.643696
 Number of u3 gates per Clifford: 0.166957
 Number of cx gates per Clifford: 1.468913

Error per gate from noise model

```
epgs_1q = {'u1': 0, 'u2': p1Q/2, 'u3': 2*p1Q/2}
epg_2q = p2Q*3/4
pred_epc = rb.rb_utils.calculate_2q_epc(
    gate_per_cliff=gates_per_cliff,
    epg_2q=epg_2q,
    qubit_pair=[0, 2],
    list_epgs_1q=[epgs_1q, epgs_1q])
```

Calculate the predicted epc

```
print("Predicted 2Q Error per Clifford: %e"%pred_epc)
```

Predicted 2Q Error per Clifford: 1.565697e-02

```
rb_opts2 = rb_opts.copy()
rb_opts2['rb_pattern'] = [[0,1]]
rb_opts2['length_multiplier'] = 1
rb_circs2, xdata2 = rb.randomized_benchmarking_seq(**rb_opts2)
```

```
noise_model2 = NoiseModel()
```

#Add T1/T2 noise to the simulation

```
t1 = 100.
t2 = 80.
gate1Q = 0.1
gate2Q = 0.5
noise_model2.add_all_qubit_quantum_error(thermal_relaxation_error(t1,t2,gate1Q), 'u2')
noise_model2.add_all_qubit_quantum_error(thermal_relaxation_error(t1,t2,2*gate1Q), 'u3')
noise_model2.add_all_qubit_quantum_error(
    thermal_relaxation_error(t1,t2,gate2Q).tensor(thermal_relaxation_error(t1,t2,gate2Q)), 'cx')
backend = qiskit.Aer.get_backend('qasm_simulator')
```

```
basis_gates = ['u1','u2','u3','cx'] # use U,CX for now
shots = 500
result_list2 = []
transpile_list2 = []
for rb_seed,rb_circ_seed in enumerate(rb_circs2):
    print('Compiling seed %d'%rb_seed)
    rb_circ_transpile = qiskit.transpile(rb_circ_seed, basis_gates=basis_gates)
    print('Simulating seed %d'%rb_seed)
    job = qiskit.execute(rb_circ_transpile, noise_model=noise_model, shots=shots, backend=backend,
backend_options={'max_parallel_experiments': 0})
    result_list2.append(job.result())
    transpile_list2.append(rb_circ_transpile)
print("Finished Simulating")
```

```
Compiling seed 0
Simulating seed 0
Compiling seed 1
Simulating seed 1
Compiling seed 2
Simulating seed 2
Compiling seed 3
Simulating seed 3
Compiling seed 4
Simulating seed 4
Finished Simulating
```

#Create an RBFitter object

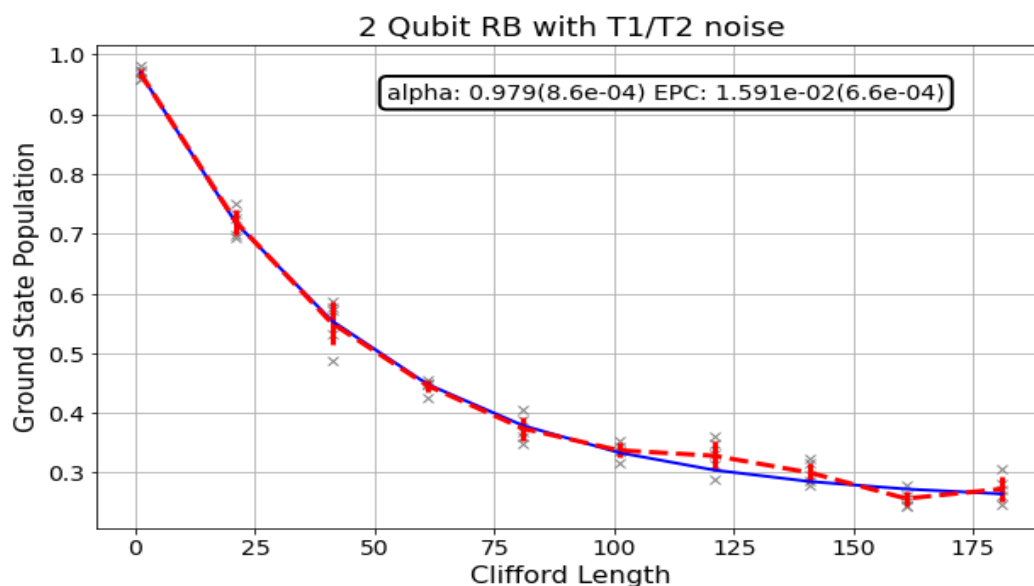
```
rbfit = rb.RBFitter(result_list2, xdata2, rb_opts2['rb_pattern'])
```

```
plt.figure(figsize=(10, 6))
ax = plt.gca()
```

```
# Plot the essence by calling plot_rb_data
rbfit.plot_rb_data(0, ax=ax, add_label=True, show_plt=False)
```

```
# Add title and label
ax.set_title('2 Qubit RB with T1/T2 noise', fontsize=18)
```

```
plt.show()
```



#Count the number of single and 2Q gates in the 2Q Cliffords

```
gates_per_cliff = rb.rb_utils.gates_per_clifford(transpile_list2,xdata[0],basis_gates,rb_opts2['rb_pattern'][0])
```

```
for basis_gate in basis_gates:
    print("Number of %s gates per Clifford: %f"%(basis_gate ,
                                                np.mean([gates_per_cliff[0][basis_gate],
                                                         gates_per_cliff[1][basis_gate]])))
```

```
Number of u1 gates per Clifford: 0.119674
Number of u2 gates per Clifford: 1.655000
Number of u3 gates per Clifford: 0.181848
Number of cx gates per Clifford: 1.500652
```

```
# Predicted primitive gate errors from the coherence limit
u2_error = rb.rb_utils.coherence_limit(1,[t1],[t2],gate1Q)
u3_error = rb.rb_utils.coherence_limit(1,[t1],[t2],2*gate1Q)
epg_2q = rb.rb_utils.coherence_limit(2,[t1,t1],[t2,t2],gate2Q)
epgs_1q = {'u1': 0, 'u2': u2_error, 'u3': u3_error}
pred_epc = rb.rb_utils.calculate_2q_epc(
    gate_per_cliff=gates_per_cliff,
    epg_2q=epg_2q,
    qubit_pair=[0, 1],
    list_epgs_1q=[epgs_1q, epgs_1q])
```

```
# Calculate the predicted epc
print("Predicted 2Q Error per Clifford: %e"%pred_epc)
```

Predicted 2Q Error per Clifford: 1.320763e-02

```
import qiskit.tools.jupyter
%qiskit_version_table
%qiskit_copyright
```

Qiskit Software Version

qiskit-terra	0.20.2
qiskit-aer	0.10.4
qiskit-ignis	0.7.1
qiskit-ibmq-provider	0.19.1
qiskit	0.36.2
qiskit-nature	0.4.1
qiskit-finance	0.3.3
qiskit-optimization	0.4.0
qiskit-machine-learning	0.4.0

System information

```
Python version      3.8.13
Python compiler GCC 10.3.0
Python build default, Mar 25 2022 06:04:10
OS                  Linux
CPUs8Memory (Gb) 31.211315155029297
Wed Jul 06 14:56:59 2022 UTC
```