Demonstrating Circuit Benchmarking for Efficient and Scalable Fidelity Estimation

by

Danesh Morales Hashemi

A project

presented to the University of Waterloo

as a partial fulfillment of the requirements of PHYS 437 B

Waterloo, Ontario, Canada

April, 2025

Abstract

Quantum computers have the capability to outperform classical computers by lowering the computational complexity of certain tasks, such as factoring large integers with Shor's algorithm [1]. Despite this potential, real-world quantum devices are affected by imperfections in their implementations.

A way to measure the effectiveness of a quantum computer when running a quantum circuit is to compute the process fidelity [2] of the circuit. The process fidelity is *the* figure of merit to quantify how accurate a quantum computer is. The process fidelity tells us how much we can *trust* the bit-string output after running a circuit. Even though calculating the process fidelity is possible, it is highly impractical, both analytically and experimentally, as we work with long depth circuits with a significant number of qubits.

In this project, we will illustrate the theoretical concept of circuit benchmarking [3], an efficient and scalable protocol to "lower bound" the process fidelity of a circuit. Finding a lower bound to the process fidelity is crucial because it tells us what is the worst-case error rate in our quantum circuit. We will exhaustively run simulations on quantum circuits with different error models and error strengths to measure the closeness of the theoretical bound imposed by the circuit benchmarking theory with respect to the actual process fidelity. We will find that in several cases there is a minimal or zero gap, and thus not only we have found a lower bound, but rather a good approximation to the process fidelity.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Definitions

The PHYS 437A project from last year is included as supplementary information in this project (3). It is advisable to review the definitions in Section 3.1 before proceeding with the PHYS 437B project.

This section introduces important definitions that were not covered in Section 3.1, and provides simplified versions of some definitions from Section 3.1 too.

Definition 1.1.1 (Process Fidelity [2])**.** Given an ideal unitary map U and noisy implementation $\mathcal{E}$, the process fidelity $\mathcal{F}(U, \mathcal{E})$ is a scalar quantity that represents the accuracy of our noisy implementation. For example, if $\mathcal{E} = U$ (which means no error in the implementation) then $\mathcal{F}(U, \mathcal{E}) = \mathcal{F}(U, U) = 1$, which is the maximum process fidelity we can attain. Mathematically, we can represent the process fidelity as

$$\mathcal{F}(U, \mathcal{E}) = \frac{1}{4^n} \operatorname{Tr}([\mathcal{E}][U]^\dagger) \tag{1.1.1}$$

where $[U], [\mathcal{E}]$ are the Liouville representations [2] of U and $\mathcal{E}$ respectively, and n is the number of qubits of the system.

Remark: We will sometimes refer to the process fidelity as "fidelity" for simplicity.

Definition 1.1.2 (Process Infidelity [2])**.**

$$r(U, \mathcal{E}) = 1 - \mathcal{F}(U, \mathcal{E}) \tag{1.1.2}$$

Definition 1.1.3 (Cycle)**.** A cycle is a set of parallel gates implemented in a quantum circuit. As a visual example of a cycle, see Figure 3.1.

Definition 1.1.4 (Cycle Benchmarking [4])**.** Cycle Benchmarking (CB) is a highly scalable and efficient error diagnostics protocol used to measure the process fidelity of a target cycle. The effectiveness of Cycle Benchmarking depends on several parameters, which are beyond the scope of this project. In general, increasing computational resources improves the accuracy of the process fidelity estimation.

It is important to emphasize the significance of Cycle Benchmarking: on a real quantum device, the process fidelity (i.e. Equation 1.1.1) cannot be used to compute the process fidelity of a quantum circuit for two main reasons:

1. The exact mathematical model of $\mathcal{E}$ is typically unknown.
2. Even if $\mathcal{E}$ could be modeled mathematically, the complexity of Equation 1.1.1 scales exponentially with the number of qubits.

Remark: The estimated process fidelity obtained after running cycle benchmarking on a cycle of interest is **not** the process fidelity of the cycle of interest itself [4], but rather the process fidelity of the composition of the cycle of interest with an "average" cycle made out of easy gates (see Definition 3.1.1). This composition is also referred as a **dressed cycle**. For the purpose of this project, obtaining the process fidelity of the dressed cycle is crucial.

1.2 Circuit Benchmarking

1.2.1 Introduction

As mentioned in the "Next Steps" section of the PHYS 437A project (page 59), the focus of the PHYS 437B project was to test theory of circuit benchmarking. Although Circuit Benchmarking was introduced in Section 4.3, we will revisit the theory to address some inaccuracies.

Remark: The Circuit Benchmarking paper [3] is still unpublished.

1.2.2 Circuit Benchmarking Example

We will illustrate circuit benchmarking using as an example a 5-qubit circuit with 21 cycles.

The randomized compiling theory [5] (refer to Section 3.2 if needed) tell us that we can rewrite any circuit as alternating rounds of easy and hard gates. Let $\vec{E}_i$ be the easy gate cycles, and $\vec{G}_i$ the hard gate cycles. The layout of our 5-qubit circuit is

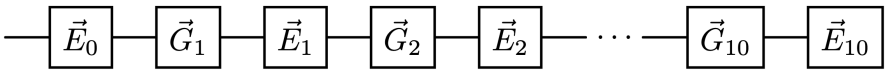


Figure 1.1: 5-qubit circuit layout

The circuit consists of 10 sequential pairs of easy and hard gate cycles $(\vec{E}_i, \vec{G}_i)$, each pair is called a dressed cycle $\mathcal{D}_i$. Additionally, we have an extra easy cycle $\vec{E}_0$ at the beginning, for a total of 21 cycles.

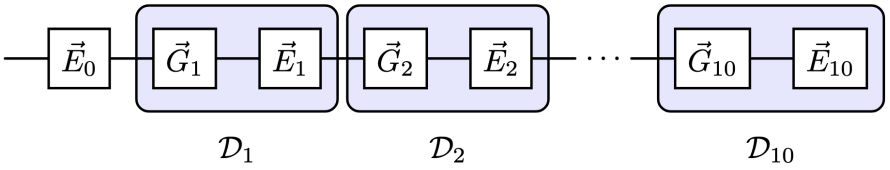


Figure 1.2: Dressed gates of the 5-qubit circuit

Let $\mathcal{F}(\mathcal{D}_i)$ be the process fidelity of a dressed gate $\mathcal{D}_i$ measured via cycle benchmarking. We can estimate the fidelity of $\vec{E}_0$ by "dressing" the cycle with an identity cycle $\mathcal{I}$; that is,

$$F(\vec{E}_0) = \mathcal{F}(\mathcal{D}_0) = \mathcal{F}(\vec{E}_0 \circ \mathcal{I}) \tag{1.2.1}$$

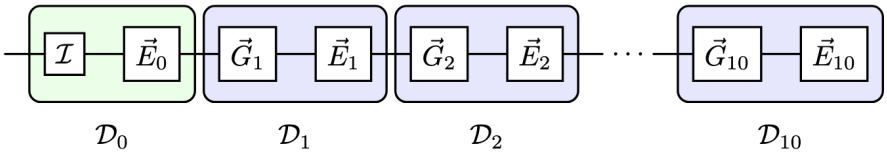


Figure 1.3: Estimating $F(\vec{E}_0)$ by dressing the easy cycle $\vec{E}_0$ with an identity cycle $\mathcal{I}$

From this perspective, we have 11 dressed cycles instead of 10. Consider the subcircuit $\mathcal{C}_l$, which is the composition of the first l dressed cycles of $\mathcal{C}$, where $1 \leq l \leq 11$, and $\mathcal{C}_{11} = \mathcal{C}$. Then $F_{\mathrm{CB}}^{\mathrm{predicted}}(\mathcal{C}_l)$ is defined as

$$F_{\mathrm{CB}}^{\mathrm{predicted}}(\mathcal{C}_l) = \prod_{i=0}^{l-1} \mathcal{F}(\mathcal{D}_i) \tag{1.2.2}$$

Similar to $F_{\text{CB}}^{\text{predicted}}$, we want to compute $F_{\text{RC}}^{\text{actual}}$, which is the fidelity of a circuit under the Randomized Compiling protocol [5], for each subcircuit $\mathcal{C}_l$. Since we don't need to dress the $\vec{E}_0$ cycle, an equivalent definition for a subcircuit $\mathcal{C}_l$ is the composition of the first $2l-1$ cycles of $\mathcal{C}$, where $1 \leq l \leq 11$, and $\mathcal{C}_{11} = \mathcal{C}$.

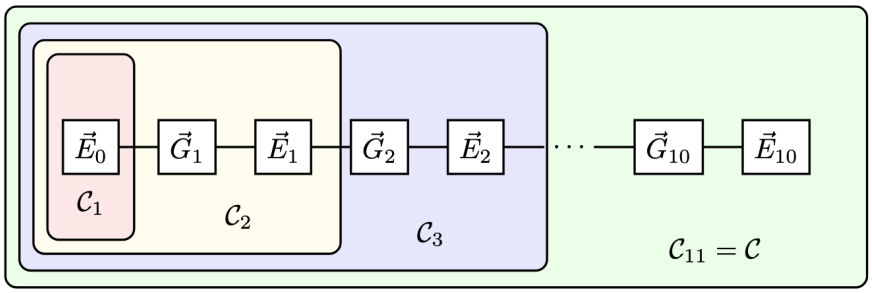


Figure 1.4: Breaking down $\mathcal{C}$ into subcircuits $\mathcal{C}_l$ to compute $F_{\text{RC}}^{\text{actual}}(\mathcal{C}_l)$

Under ideal assumptions (i.e the noise follows a Markovian error model and appropriate error models/infidelities for the easy and hard gates, etc.) the following inequality holds [3]:

$$F_{\text{RC}}^{\text{actual}}(\mathcal{C}_l) \geq F_{\text{CB}}^{\text{predicted}}(\mathcal{C}_l) \tag{1.2.3}$$

The theory suggests that the gap between the two quantities of interest is relatively small. In Section 2.4, we will plot $F_{\text{RC}}^{\text{actual}}(\mathcal{C}_l)$ and $F_{\text{CB}}^{\text{predicted}}(\mathcal{C}_l)$ for $l \in \{1, \ldots, 11\}$ with different noise settings to visualize how the gap fluctuates, and detect if the theory breaks down.

1.3 Possible Questions About Circuit Benchmarking

In this section, we will answer some important questions that the reader might have regarding circuit benchmarking.

1. *Circuit benchmarking finds a lower bound of the process fidelity of a circuit under randomized compiling, instead of the original circuit. Why do we care about the process fidelity of a circuit under randomized compiling?*

   Answer: Let C be an ideal quantum circuit, and C_{RC} the circuit under randomized compiling. Then $C \equiv C_{RC}$. Recall that randomized compiling does not alter the ideal circuit; it removes the coherent errors from the noisy implementation [5]. That is, $\mathcal{E}(C) \not\equiv \mathcal{E}(C_{RC})$. Therefore, even though $C \equiv C_{RC}$, then $\mathcal{F}(\mathcal{E}(C)) \neq \mathcal{F}(\mathcal{E}(C_{RC}))$. This is perfectly fine. Randomized compiling is widely implemented in real quantum devices [6], so it is expected by default that quantum circuits are randomly compiled. Thus, finding a lower bound on $\mathcal{F}(\mathcal{E}(C_{RC})$ instead of $\mathcal{F}(\mathcal{E}(C))$ is essential.

2. *Why is circuit benchmarking "efficient and highly scalable"?*

   Answer:

   (a) We can efficiently estimate $\mathcal{F}(\mathcal{D}_l)$ via cycle benchmarking for any hard dressed cycle of interest to estimate $F_{\text{CB}}^{\text{predicted}} = \prod_{l=1}^{L} \mathcal{F}(\mathcal{D}_l)$, independent of state preparation/measurement errors and number of qubits [4].

   (b) Given a circuit C with L_{unique} hard cycles, and $L_{\text{unique}} \leq L$, then we only need to run cycle benchmarking $L_{\text{unique}} + 1$ times to estimate $F_{\text{CB}}^{\text{predicted}}$ [3]. This is because if there are repeated hard cycles in our circuit, we only need to run cycle benchmarking once per unique cycle. The $+ 1$ cycle benchmarking run is to estimate the fidelity of the unpaired easy cycle, as shown in Figure 1.3.

Chapter 2

Experiments and Results

2.1 Introduction

The experiments and results for this project were obtained using TrueQ [7]. TrueQ is a Python library that can simulate quantum circuits and protocols such as randomized compiling and cycle benchmarking.

2.2 Randomized Compiling

2.2.1 Impact of a Perfect Pauli Twirl via RC on the Fidelity of Quantum Circuits

Figure 2.1 examines how the process fidelity is affected by the number of hard gate cycles, and compares the bare fidelity of randomly generated circuits with their fidelity under RC, assuming a perfect Pauli twirl has been implemented on the noise channels.

Each data point represents the process fidelity of one of 100 independently generated 5-qubit circuits for each hard cycle count. The noise model applied is an over-rotation error (4.1.1) on the hard gates with strength equal to 0.1, while the easy gates remain

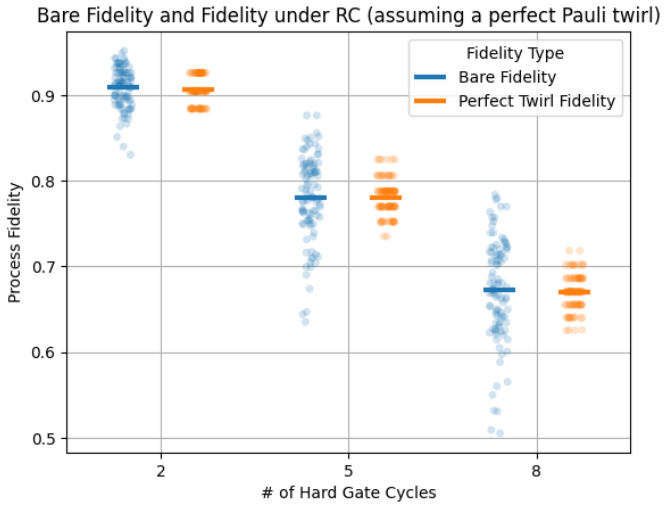


Figure 2.1: Comparing Bare Fidelity and RC Fidelity with a Perfect Twirl

noiseless. The blue points correspond to bare fidelity, while the orange points represent fidelity under RC with a perfect Pauli Twirl. Hard cycles consist of either (2 CNOTs + 1 H) or (2 CNOTs + 1 T). The hard gates act on random qubits, ensuring that each qubit is affected exactly by one hard gate. The figure suggests that both fidelities decrease as the number of hard cycles increases. Additionally, the range of fidelities under RC is narrower. This behaviour is expected, according to the theory from [5] and [8].

2.2.2 Fidelity Comparison between Perfect Pauli and Finite Randomizations

As mentioned in Section 3.2, running RC with finite randomizations will approximate the error channels that act on the quantum circuit, to Pauli error channels (3.1.7), as the action of averaging the randomizations has a similar behaviour as Pauli twirling (3.1.8) the noise channels.

Although it is possible to perform a perfect Pauli twirl with carefully chosen finite number of randomizations, it is inefficient; the number of randomizations scales exponentially with the number of qubits. Nevertheless, it has been shown that we can approximate a

Pauli twirl with a fixed number of randomizations, regardless of the number of qubits, and depth of the circuit [5]. As an example, Figure 2.2 illustrates the accuracy of running RC with a finite number of randomizations by comparing the fidelity to that under a perfect Pauli twirl of randomly generated 5-qubit circuits with varying numbers of hard cycles, each undergoing 100 randomizations under RC.

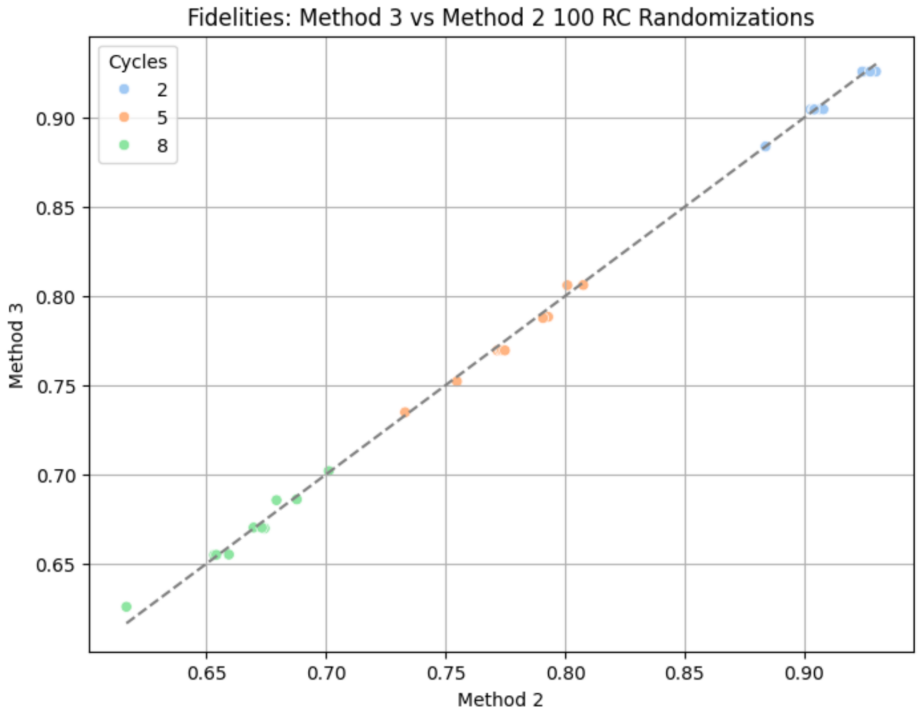


Figure 2.2: Fidelities under a Perfect (Method 2) vs Approximate (Method 3) Pauli Twirl

Each point represents the fidelity of a single 5-qubit circuit from a set of 10 random bare circuits for each hard cycle count. The close alignment of the data points to the $y = x$ line suggests that the fidelity of the bare circuits under 100 randomizations closely approximates the fidelity under a perfect Pauli twirl.

2.3 Testing Circuit Benchmarking Code

2.3.1 Comparing Custom Code with QCAP

There are two ways to compute $F_{\text{CB}}^{\text{predicted}}$: Using a custom made by the student (Danesh), and a function named **QCAP**, provided by TrueQ. Our goal is to determine whether the custom code and QCAP agree. To test this, we will run both methods on a single 2-qubit circuit with two hard cycles (single CNOT in each hard cycle) with two different number of randomizations: 100 and 1000. We will run both methods for 10 iterations on the same circuit, and compute the average fidelity and STD from the 10 iterations.

Custom Code		QCAP	
Fidelity	STD	Fidelity	STD
0.96359	0.00114	0.96223	0.00130
0.96286	0.00117	0.96445	0.00117
0.96578	0.00106	0.96488	0.00109
0.96329	0.00115	0.96396	0.00118
0.96305	0.00117	0.96723	0.00099
0.96386	0.00114	0.96608	0.00106
0.96329	0.00111	0.96016	0.00127
0.96280	0.00117	0.96372	0.00113
0.95931	0.00147	0.96198	0.00117
0.96441	0.00111	0.96601	0.00108

Table 2.1: Comparison of Custom Code with QCAP with 100 randomizations

Custom Code		QCAP	
Fidelity	STD	Fidelity	STD
0.96322	0.00117	0.96407	0.00114

Table 2.2: Average values from Table 2.1

We can observe that the average difference between the fidelities is ≈ 0.0085 and between the STDs is ≈ 0.00003. The possible explanations for this discrepancy are that there were not enough randomizations, and that the randomizations for each case were different. We will replicate the same test, but with 1000 randomizations instead of 100.

Custom Code		QCAP	
Fidelity	STD	Fidelity	STD
0.96291	0.00035	0.96376	0.00035
0.96316	0.00035	0.96435	0.00035
0.96363	0.00035	0.96368	0.00036
0.96341	0.00035	0.96399	0.00035
0.96379	0.00034	0.96338	0.00036
0.96347	0.00035	0.96399	0.00036
0.96340	0.00035	0.96324	0.00036
0.96371	0.00034	0.96303	0.00037
0.96321	0.00035	0.96352	0.00036
0.96347	0.00035	0.96364	0.00035

Table 2.3: Comparison of Custom Code with QCAP with 1000 randomizations

Custom Code		QCAP	
Fidelity	STD	Fidelity	STD
0.96342	0.00035	0.96366	0.00036

Table 2.4: Average values of Table 2.3

With 1000 randomizations, the average difference between the fidelities is ≈ 0.00024, and between the STDs is ≈ 0.00001. Comparing Table 2.1 with Table 2.3, we can see that increasing the randomizations brings the two fidelities closer together. Additionally, the STDs decreased by an order of magnitude.

From the numerical evidence, it is clear that there is no significant difference using the custom code or QCAP. However, the run-time of QCAP is approximately 40% faster. Therefore, QCAP is a more viable option.

2.4 Circuit Benchmarking Experiment

2.4.1 Introduction

The plots presented in this section numerically illustrate the circuit benchmarking protocol on a single 5-qubit quantum circuit $\mathcal{C}$ with varying strength of over-rotation (gate dependent) errors (see 4.1.1 for reference) on both the easy and hard gates. The 5-qubit circuit has 21 cycles: 11 easy cycles and 10 hard cycles (see Figure 2.3). The easy cycle gates were randomly chosen from a uniform distribution of the easy gate set defined in Section 3.1. In contrast, each hard cycle contains exactly 2 CNOT gates and one single-qubit gate, either a H or T gate, placed randomly across the 5 qubits, always ensuring that each qubit is acted upon by precisely one gate. In the particular circuit we analyze here, there are 5 hard cycles with H gates and 5 cycles with T gates. The order of the 10 hard cycles was randomly chosen. We will evaluate the Actual Fidelity under RC, denotes as $F_{\text{RC}}^{\text{actual}}$ and the Predicted Lower Bound Fidelity via CB, denoted as $F_{\text{CB}}^{\text{predicted}}$ for different subcircuits extraced from $\mathcal{C}$ providing insight on how the fidelity decays as the subcircuit gets bigger (i.e as the number of cycles increase).

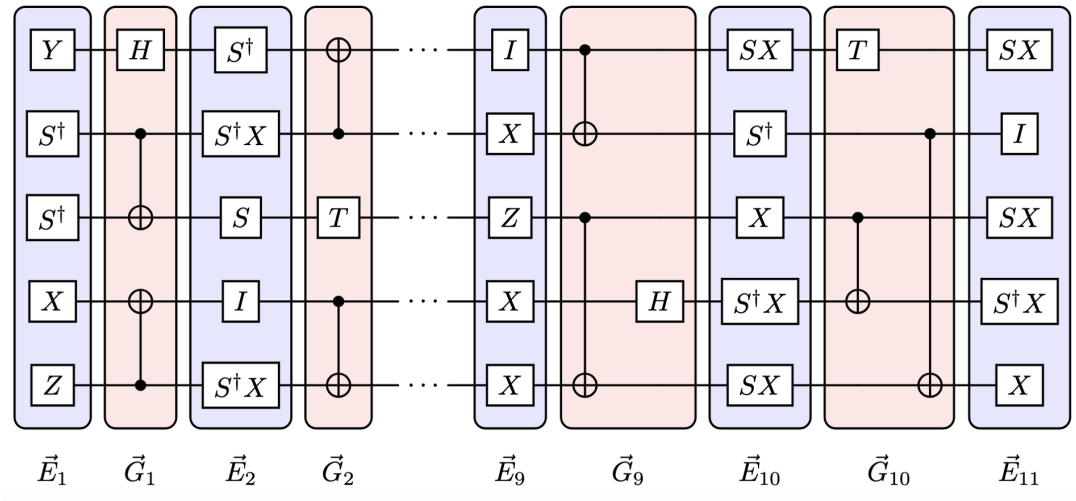


Figure 2.3: Circuit used for the experiments

2.4.2 Error-Free Easy Gates

For this plot, the easy gates are perfectly implemented, and each hard gate has an overrotation error of strength = 0.1.

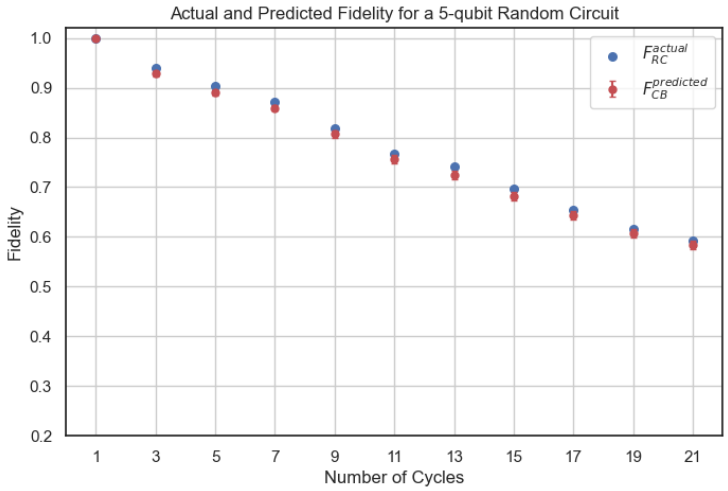


Figure 2.4: Circuit Benchmarking with error-free easy gates

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0245	0.0015	0.0183	0.0148	0.0489

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.000	0.000	0.000

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.000	0.000

Table 2.5: Infidelities from Figure 2.4

Figure 2.4 shows that $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ are extremely close to each other, while still satisfying the cycle benchmarking inequality given in Equation 1.2.3.

2.4.3 Noisy Easy Gates: Part 1

The over-rotation strength for the hard gates remains unchanged from the previous plot; however, we have introduced an over-rotation to the easy gates, such that the average infidelity of an easy gate is approximately a third that of a hard gate.

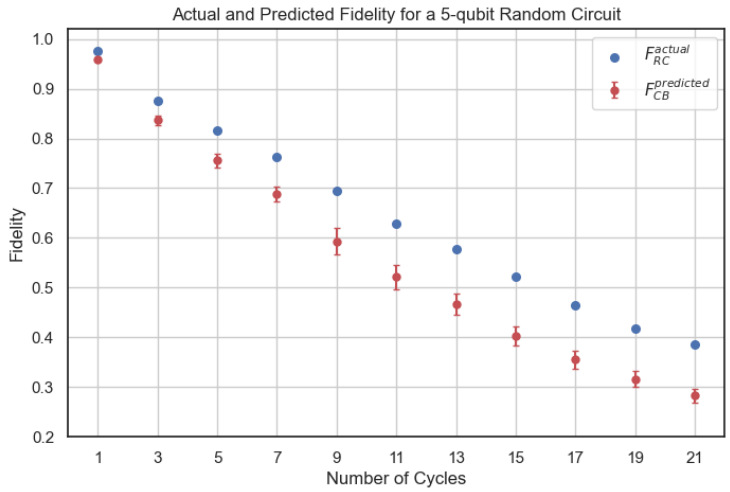


Figure 2.5: Adding errors to the easy gates (Easy/Hard Gate Infidelity Ratio = 1/3)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0245	0.0015	0.0183	0.0148	0.0489

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0049	0.0031	0.0244

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.3337	0.4996

Table 2.6: Infidelities from Figure 2.5

By introducing noise to the easy gates, the discrepancy between $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ in Figure 2.5 increased. In the following plots, the over-rotation error strength of the easy gates will be varied while maintaining the same error models for the hard gates. This will allow us to examine how the gap between $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ evolves as the fidelity of the easy gates changes.

2.4.4 Noisy Easy Gates: Part 2

We will start decreasing the infidelity of the easy gates. In Figure 2.5, the ratio between the average infidelities of the easy and hard gates was $\approx 1/3$. In Figure 2.6, we adjust the strength of the over-rotation error on the easy gates so that the ratio is $\approx 1/10$.

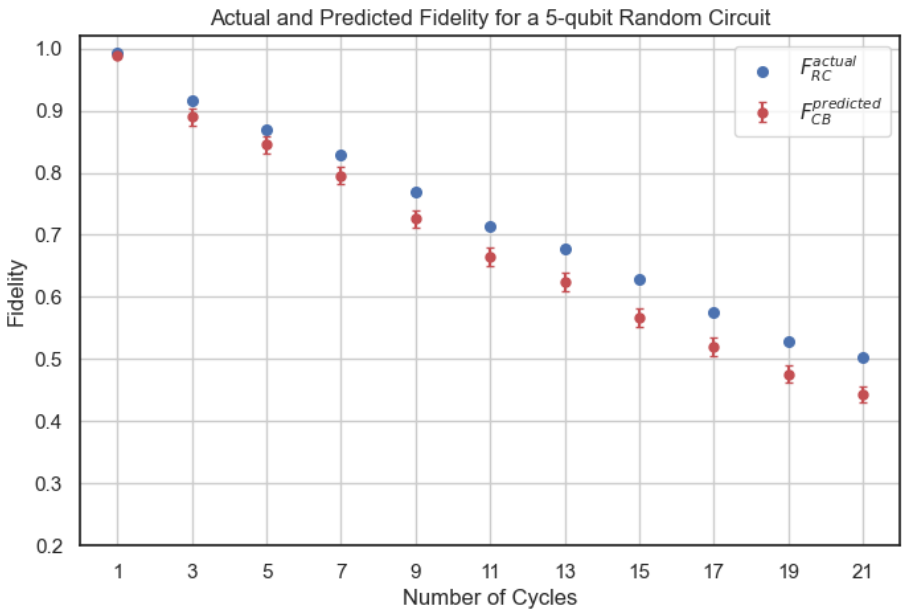


Figure 2.6: Adding errors to the easy gates (Easy/Hard Gate Infidelity Ratio = 1/10)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0245	0.0015	0.0183	0.0148	0.0489

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.00147	0.0009	0.0074

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.0997	0.1504

Table 2.7: Infidelities from Figure 2.6

2.4.5 Noisy Easy Gates: Part 3

In Figure 2.7, the ratio between the average easy and hard gates is $\approx 1/50$

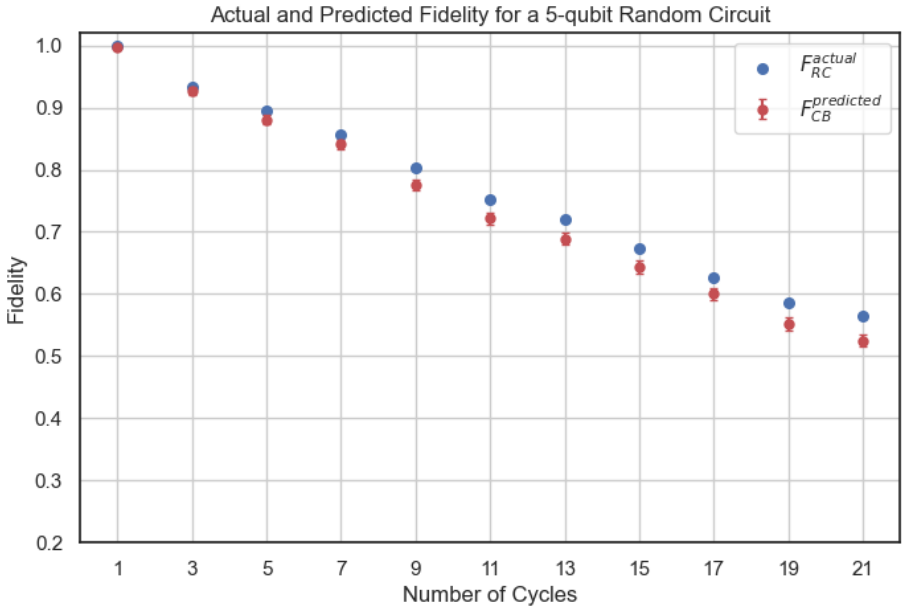


Figure 2.7: Adding errors to the easy gates (Easy/Hard Gate Infidelity Ratio = 1/50)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0245	0.0015	0.0183	0.0148	0.0489

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0003	0.0002	0.0015

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.0200	0.0302

Table 2.8: Infidelities from Figure 2.7

From Figures 2.4, 2.5, 2.6, 2.7, we can observe that the higher the infidelity the easy gates have, the steeper the decay for both fidelities is; and not only that, the gap between $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ is larger as the infidelity of the easy gates increases. We can also see that the variance of the easy gates infidelities decreases as the easy gate average infidelity gets smaller.

2.4.6 Increasing the hard gate over-rotations while keeping the easy gate infidelities fixed: Part 1

In Figure 2.5, we can see that the gap between the two fidelities is larger than in Figure 2.4, when the easy gates have no error. We are trying to answer the following question: *"Is the gap due to the ratio of the average easy and hard cycle fidelities? Or is it due to the fact that the easy cycle fidelities has a larger variation as the easy gate error rate increases?"* To answer this question, we will do the following: Keep the easy cycle infidelity fixed, with the current gate-dependent over-rotation error model, but look at a sequence of increasing hard gate infidelities, so the ratio increases but the variation in the easy gate cycles is constant across the sequence. We start with Figure 2.8, where the infidelities of all the hard gates are the same, 0.0183. The over-rotation error strength for the easy gates was chosen so that the ratio between the average easy and hard cycle infidelities is $= 1/2$, and we will keep this error model for Figures 2.9 and 2.10 too.

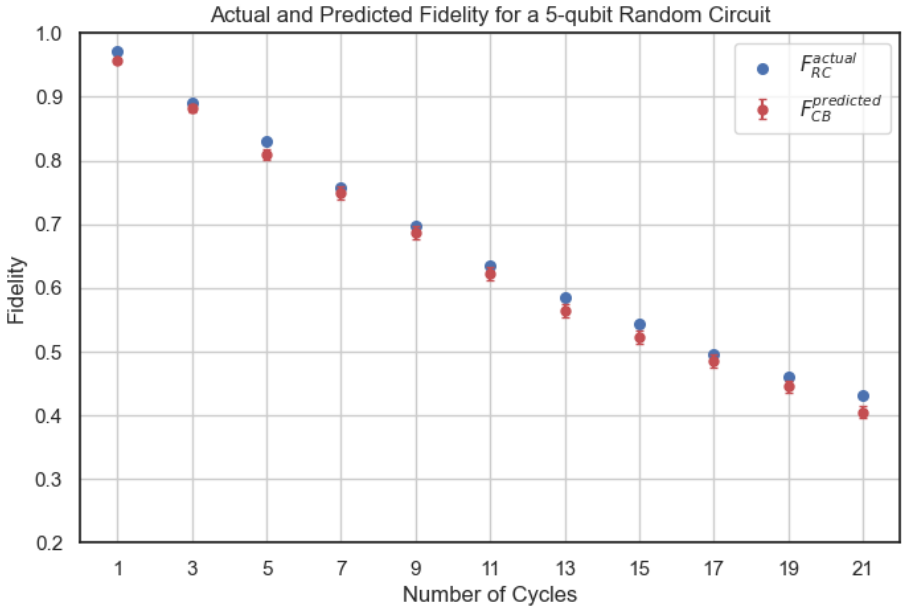


Figure 2.8: Fixed easy gate error (Avg Hard Gate Infidelity = 0.0183)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0183	0.0183	0.0183	0.0183	0.0540

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0042	0.0027	0.0209

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.2978	0.5000

Table 2.9: Infidelities from Figure 2.8

2.4.7 Increasing the hard gate over-rotations while keeping the easy gate infidelities fixed: Part 2

We increase the average hard gate infidelities from 0.0183 to 0.0236.

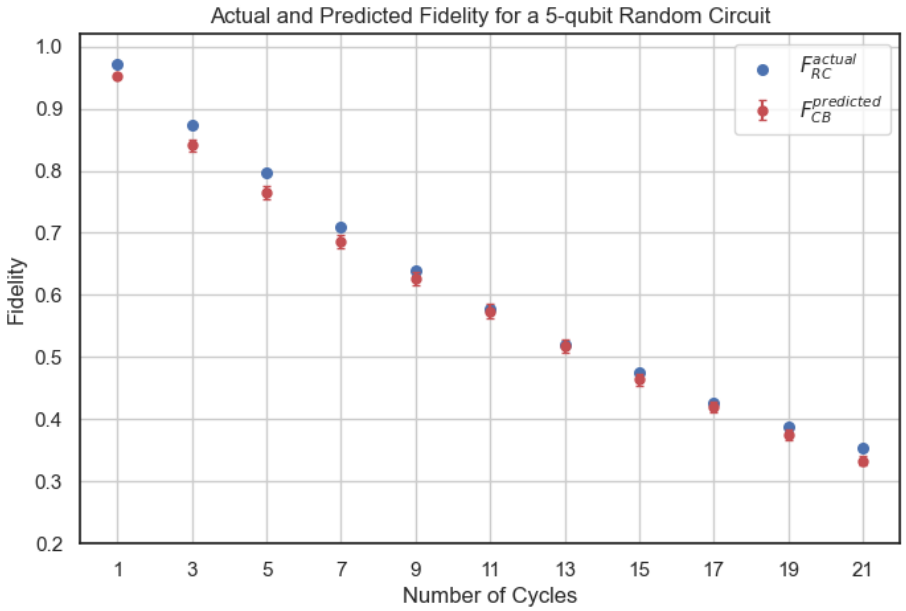


Figure 2.9: Fixed easy gate error (Avg Hard Gate Infidelity = 0.0236)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0199	0.0245	0.0263	0.0236	0.0730

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0042	0.0027	0.0209

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.2320	0.3703

Table 2.10: Infidelities from Figure 2.9

2.4.8 Increasing the hard gate over-rotations while keeping the easy gate infidelities fixed: Part 3

We increase the average hard cycle infidelity from 0.0236 to 0.0321

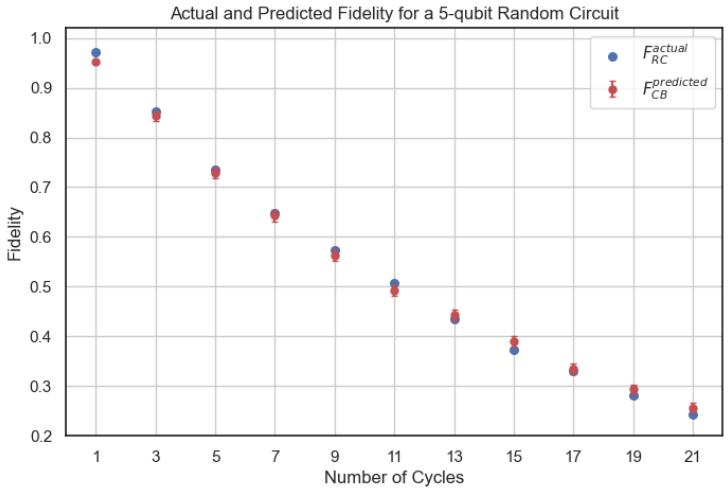


Figure 2.10: Fixed easy gate error (Avg Hard Gate Infidelity = 0.0321)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0245	0.03090	0.04087	0.0321	0.1055

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0042	0.0027	0.0209

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.1309	0.1981

Table 2.11: Infidelities from Figure 2.10

From Figures 2.8, 2.9, and 2.10, we can see that $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ decrease as the infidelities of the hard gates grow. However, the gap between $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ does not appear to increase, in contrast to Figures 2.4, 2.5, 2.6 and 2.7, where the gap increased with the infidelity of the easy gates. This suggest that the discrepancy of $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ depends on the variance of easy gate infidelities, and not on the ratio of infidelities between the easy and hard gates.

2.4.9 Adjusting the T and H over-rotations

For a fixed over-rotation error model, the T gate has an extremely small infidelity, even lower than the average easy gate; whereas, the H gate has a high infidelity, even larger than the CNOT gate. For a physical implementation, this is inaccurate. In this graph, we have tailored the individual over-rotation strengths of the H and T gates, so that the H gate infidelity is comparable to the average easy gate infidelity, and the T gate infidelity similar to that of the CNOT.

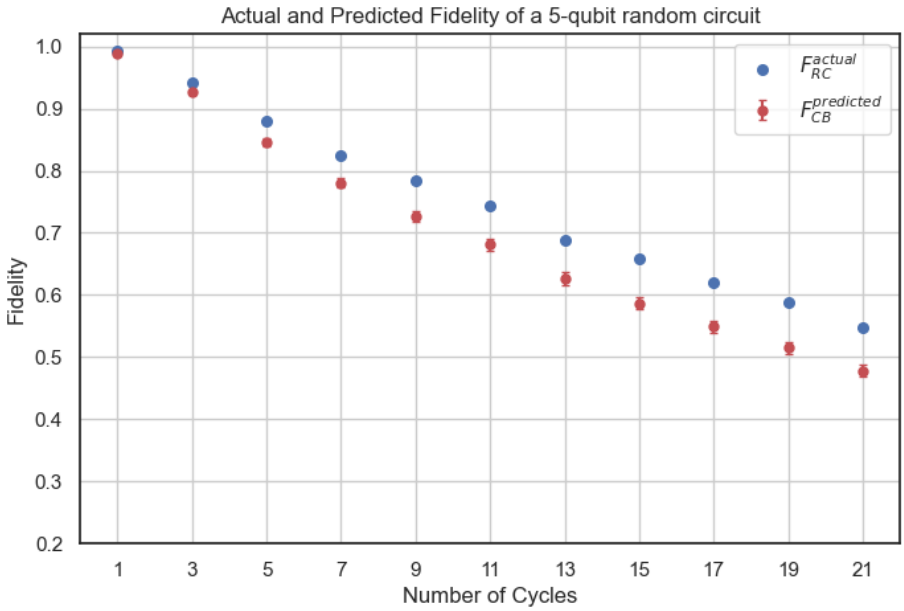


Figure 2.11: Adjusting the T and H over-rotations

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0015	0.01835	0.01835	0.01275	0.04595

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0015	0.0009	0.0074

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.1157	0.1600

Table 2.12: Infidelities from Figure 2.11

2.4.10 Comparison: Increasing the H gate infidelity

We will increase the infidelity of H so that it matches the infidelities of T, CNOT, while keeping the error rates of the other gates same as in Figure 2.11.

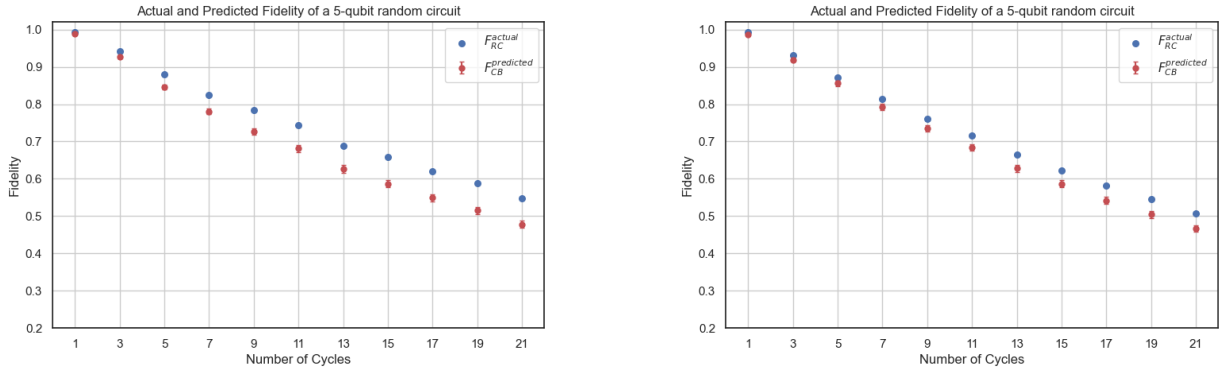


Figure 2.12: Figure 2.11 (left) vs increasing the H gate infidelities (right)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0015 $\rightarrow$ 0.01835	0.01835	0.01835	0.01275 $\rightarrow$ 0.01835	0.04595 $\rightarrow$ 0.05405

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0015	0.0009	0.0074

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.1157 $\rightarrow$ 0.0804	0.1600 $\rightarrow$ 0.1360

Table 2.13: Infidelities from Figure 2.12

Key observations:

1. $F_{\text{CB}}^{\text{predicted}}$ did not have noticeable changes.

2. $F_{\text{RC}}^{\text{actual}}$ decreased, thus closing the gap.

2.4.11 Comparison: Decreasing the infidelity of the easy gates

We will decrease the infidelity of the easy gates while keeping the error rates of the other gates the same as in Figure 2.11.

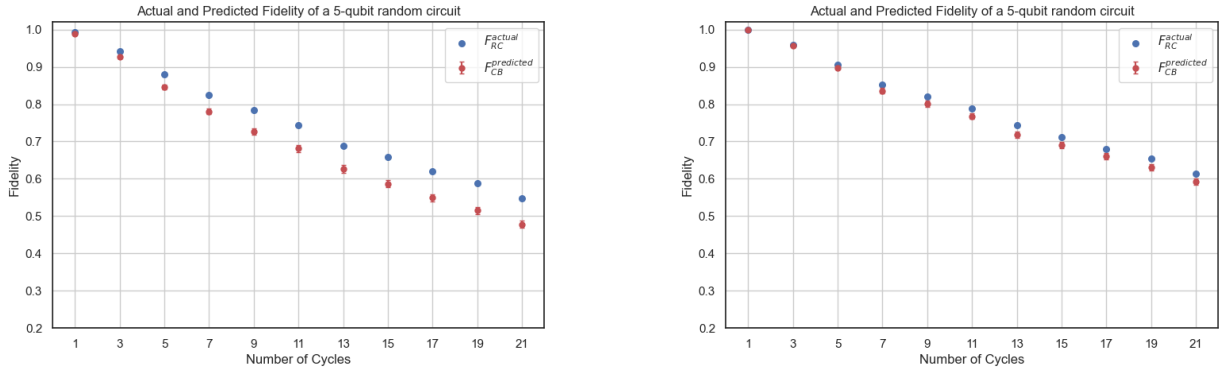


Figure 2.13: Figure 2.11 (left) vs decreasing the easy gate infidelties (right)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0015	0.01835	0.01835	0.01275	0.04595

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	$0.0015 \rightarrow 0.00017$	$0.0009 \rightarrow 0.0001$	$0.0074 \rightarrow 0.0008$

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
$0.1157 \rightarrow 0.0133$	$0.1600 \rightarrow 0.0184$

Table 2.14: Infidelities from Figure 2.13

Key observation:

1. Both $F_{\text{CB}}^{\text{predicted}}$ and $F_{\text{RC}}^{\text{actual}}$ increased, but $F_{\text{CB}}^{\text{predicted}}$ more than $F_{\text{RC}}^{\text{actual}}$, thus closing the gap.

2.4.12 Comparison: Decreasing the infidelity of the easy gates AND H gate

We will decrease both the infidelity of the easy gates and H so that they are roughly the same while keeping the error rates of the other gates the same as in Figure 2.11.

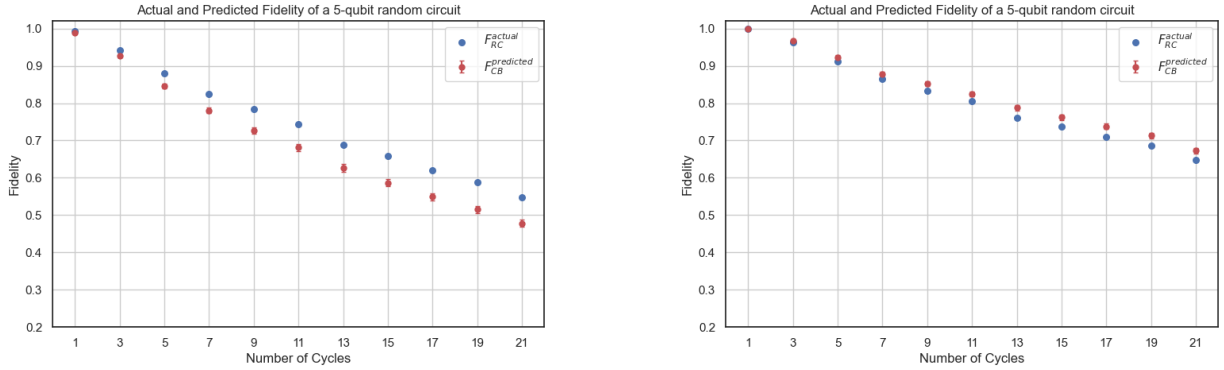


Figure 2.14: Figure 2.11 (left) vs decreasing the easy and H gate infidelities (right)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0015 $\rightarrow$ 0.000178	0.01835	0.01835	0.01275 $\rightarrow$ 0.0123	0.04595 $\rightarrow$ 0.0452

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0015 $\rightarrow$ 0.00017	0.0009 $\rightarrow$ 0.0001	0.0074 $\rightarrow$ 0.0008

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.1157 $\rightarrow$ 0.0138	0.1600 $\rightarrow$ 0.0187

Table 2.15: Infidelities from Figure 2.14

Key observation:

1. Both $F_{\text{CB}}^{\text{predicted}}$ and $F_{\text{RC}}^{\text{actual}}$ increased, but $F_{\text{CB}}^{\text{predicted}}$ more than $F_{\text{RC}}^{\text{actual}}$, which broke the inequality $F_{\text{RC}}^{\text{actual}} {}_{\text{CB}}^{\text{predicted}}$.

2.4.13 Comparison: Increasing the infidelity of the easy gates

We will increase the infidelity of the easy gates while keeping the error rates of the other gates the same as in Figure 2.11.

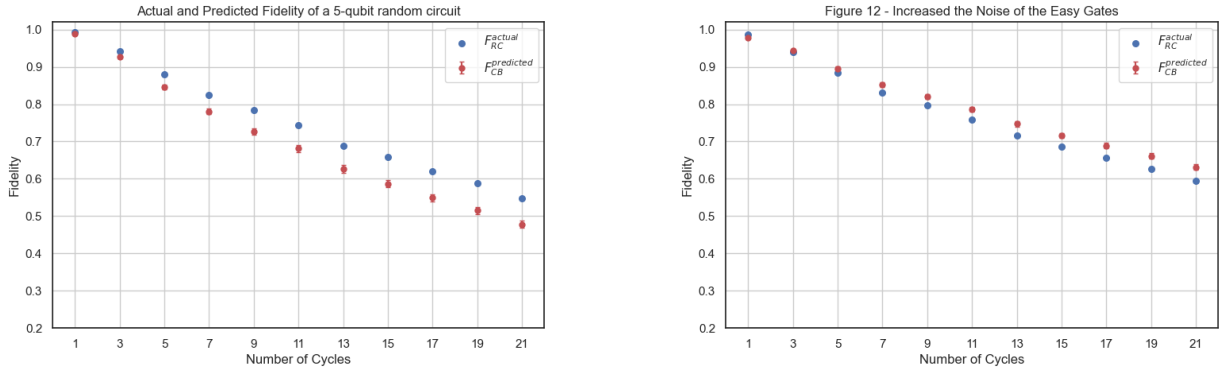


Figure 2.15: Figure 2.11 (left) vs increasing the easy gate infidelities (right)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0015	0.01835	0.01835	0.01275	0.04595

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	$0.0015 \rightarrow 0.0027$	$0.0009 \rightarrow 0.0017$	$0.0074 \rightarrow 0.0135$

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
$0.1157 \rightarrow 0.2126$	$0.1600 \rightarrow 0.2934$

Table 2.16: Infidelities from Figure 2.15

Key observations:

1. Both $F_{\text{CB}}^{\text{predicted}}$ and $F_{\text{RC}}^{\text{actual}}$ increased, but $F_{\text{CB}}^{\text{predicted}}$ more than $F_{\text{RC}}^{\text{actual}}$, which broke the inequality $F_{\text{RC}}^{\text{actual}} \geq F_{\text{CB}}^{\text{predicted}}$.

2. Interestingly, increasing the infidelity of the easy gates made both $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ increase.

2.4.14 Comparison: Decreasing the infidelities of the T and CNOT gates

We will decrease the infidelities of the T and CNOT gates while keeping the error rates of the other gates the same as in Figure 2.11.

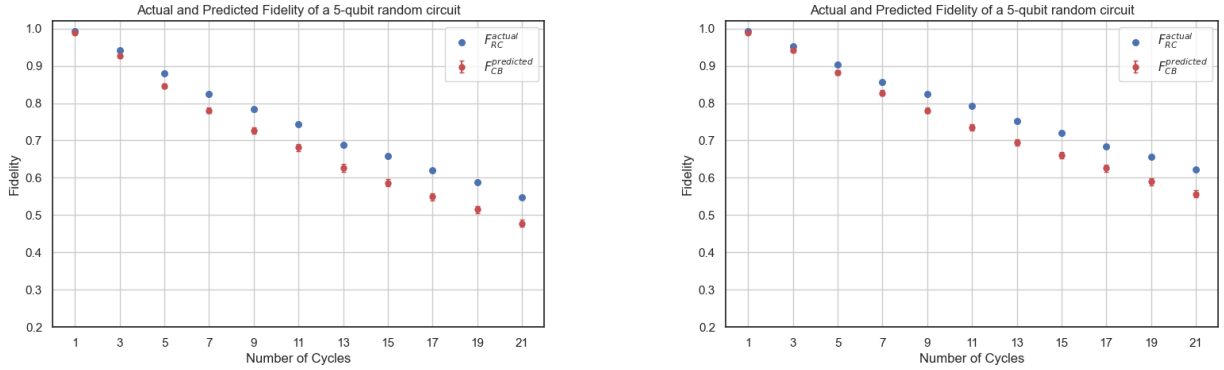


Figure 2.16: Figure 2.11 (left) vs decreasing the T and CNOT gate infidelities (right)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0015	0.01835 $\rightarrow$ 0.01382	0.01835$\rightarrow$ 0.01329	0.01275 $\rightarrow$ 0.0095	0.04595 $\rightarrow$ 0.0339

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0015	0.0009	0.0074

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.1157 $\rightarrow$ 0.1545	0.1600 $\rightarrow$ 0.2171

Table 2.17: Infidelities from Figure 2.16

Key observation:

1. Both $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ increased at the same rate, so the gap did not close, even after reducing the infidelity of T and CNOT. This might have to do with the fact that even after decreasing the errors of T and CNOT, their infidelities are still high compared to the easy + H gates.

2.4.15 Comparison: Increasing the variance of the easy gates while keeping the error rate fixed

We will increase the variance of the easy gates while keeping the same error rate by increasing the over-rotation in the 4 Paulis and decreasing the over-rotation in the 4 non-Paulis.

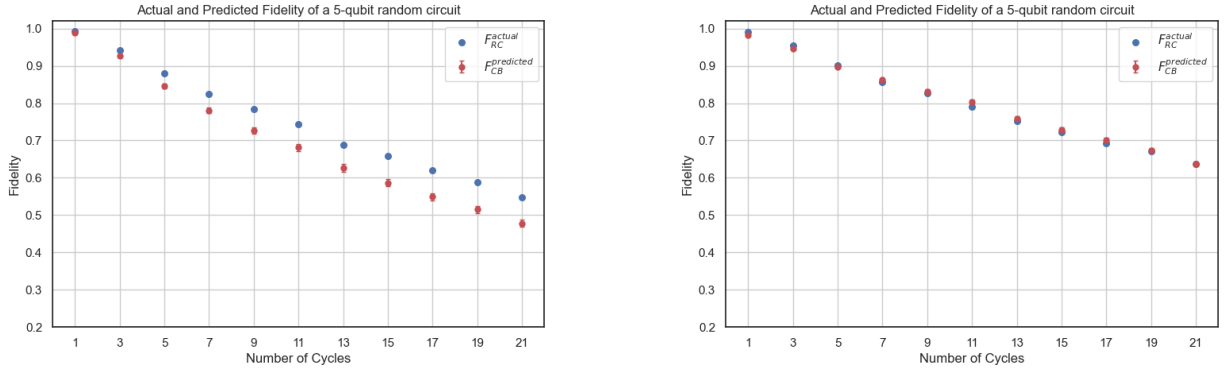


Figure 2.17: Figure 2.11 (left) vs increasing the variance of the easy gates without (right)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0015	0.01835	0.01835	0.01275	0.04595

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0015	$0.0009 \rightarrow 0.0015$	0.0074

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.1157	0.1600

Table 2.18: Infidelities from Figure 2.17

Key observation

1. By increasing the variance, both $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ increased. But $F_{\text{CB}}^{\text{predicted}}$ increased more than $F_{\text{RC}}^{\text{actual}}$, thus closing the gap. Although it seems like $F_{\text{RC}}^{\text{actual}} {}_{\text{CB}}^{\text{predicted}}$ in the right side Figure, which breaks the theory.

2.4.16 Comparison: Increasing the variance of the easy gates even more while keeping the error rate fixed

We will increase the variance of the easy gates even more while keeping the same error rate by increasing the over-rotation in the 4 Paulis and setting the non-Pauli errors to 0

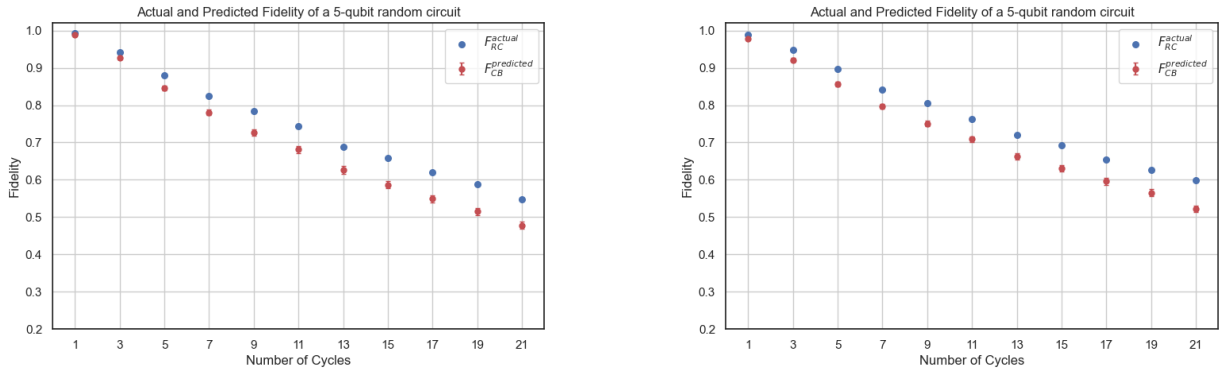


Figure 2.18: Figure 2.11 (left) vs increasing the variance of the easy gates (right)

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0015	0.01835	0.01835	0.01275	0.04595

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0015	0.0009 $\rightarrow$ 0.002	0.0074

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.1157	0.1600

Table 2.19: Infidelities from Figure 2.17

Key observation:

1. Increasing the variance even more made $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ increase at the same rate, thus the gap did not close.

2.4.17 Gate independent errors on the easy and H gates: Part 1

We change the error model of the easy and H gates to be the same fixed Z rotation error. The errors for the T and CNOT are still an over-rotation error. In this first figure, the Z error was tailored such that the easy/hard gate ratio is $\approx 1/3$.

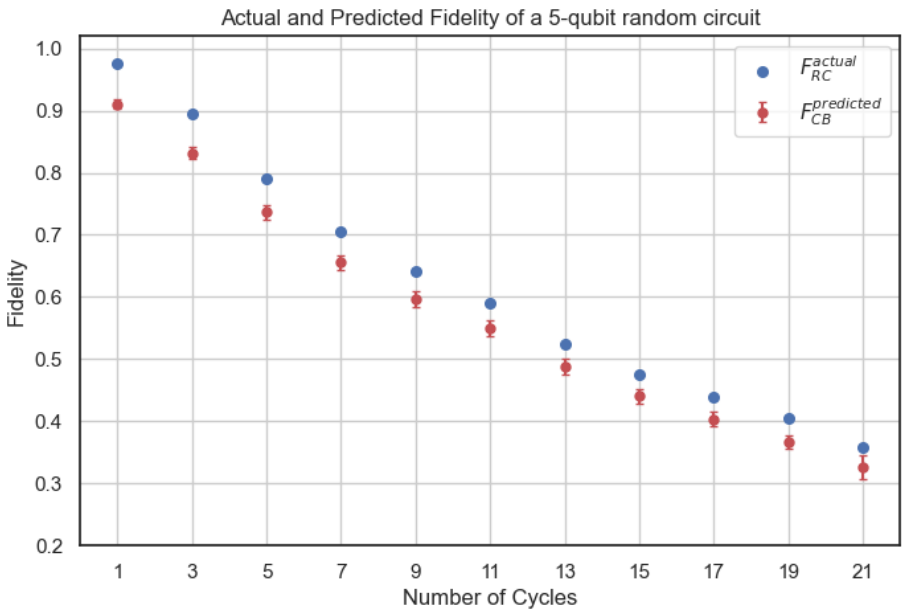


Figure 2.19: Gate independent error on easy + H gates: 1/3 error rate ratio

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0049	0.01835	0.01835	0.01386	0.0476

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0049	0.0000	0.0241

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.3511	0.5067

Table 2.20: Infidelities from Figure 2.19

2.4.18 Gate independent errors on the easy and H gates: Part 2

In this second figure, the Z error was tailored such that the easy/hard gate ratio is $\approx 1/10$ while keeping the same errors for the T and CNOT.

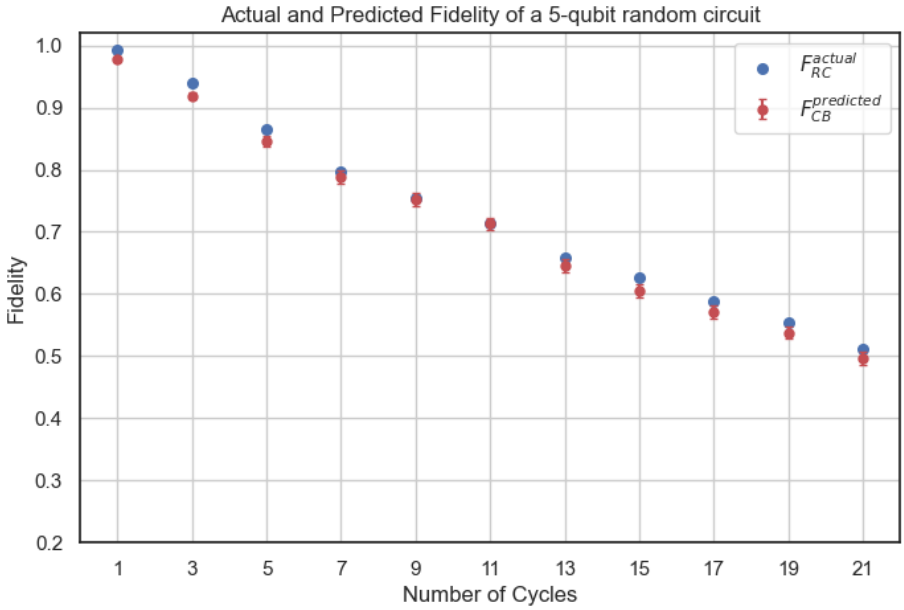


Figure 2.20: Gate independent error on easy + H gates: 1/10 error rate ratio

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.0013	0.01835	0.01835	0.0127	0.0458

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.0013	0.0000	0.0064

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.1011	0.1393

Table 2.21: Infidelities from Figure 2.20

Key observation:

1. Compared to Figure 2.19, the gap got closer by decreasing the Z rotation on the easy + H gates. Also, of course, $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ increased as there is less errors in the easy + H gates.

2.4.19 Gate independent errors on the easy and H gates: Part 3

In this third figure, the Z error was tailored such that the easy/hard gate ratio was $\approx 1/100$, while keeping the same errors for the T and CNOT.

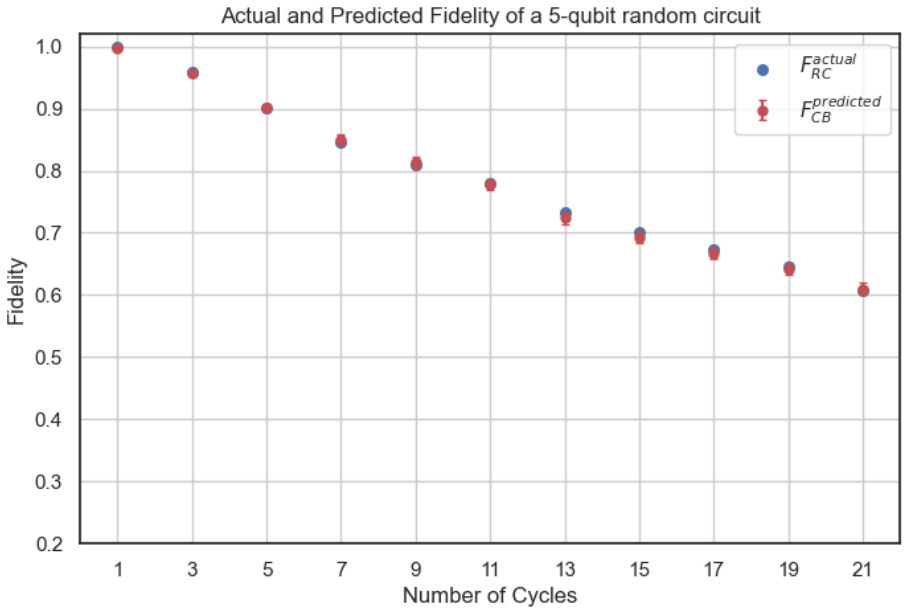


Figure 2.21: Gate independent error on easy + H gates: 1/100 error rate ratio

	H	T	CNOT	Hard Gate Avg	Hard Cycle Avg
Infidelity	0.00013	0.01835	0.01835	0.0123	0.0453

	Easy Gate Avg	Easy Gate STD	Easy Cycle Avg
Infidelity	0.00013	0.0000	0.0006

Easy/Hard Gate Avg Infidelity Ratio	Easy/Hard Cycle Avg Infidelity Ratio
0.0105	0.0142

Table 2.22: Infidelities from Figure 2.21

Key observation:

1. Compared to Figures 2.19 and 2.21, The gap between $F_{\text{RC}}^{\text{actual}}$ and $F_{\text{CB}}^{\text{predicted}}$ closed even more by significantly decreasing the infidelity of the easy gates.

Concluding Remarks

1. Circuit benchmarking does an excellent job at lower bounding the process fidelity of a quantum circuit under randomized compiling.

2. The gap between $F_{\mathrm{RC}}^{\mathrm{actual}}$ and $F_{\mathrm{CB}}^{\mathrm{predicted}}$ might be tight or loose depending on the noise model/strength.

3. Particularly, if the easy gates have a low error rate, then the gap $F_{\mathrm{RC}}^{\mathrm{actual}}$ and $F_{\mathrm{CB}}^{\mathrm{predicted}}$ is close to zero.

4. In extreme cases, where the easy gates have high infidelity and a gate dependent error model, the circuit benchmarking inequality might break.

5. When the error model on the easy gates is gate independent, the inequality does not break, regardless of the infidelity of the easy gates.

6. There *might* be a correlation between the size of the gap between $F_{\mathrm{RC}}^{\mathrm{actual}}$ and $F_{\mathrm{CB}}^{\mathrm{predicted}}$ and the variance of the infidelities of the easy gates.

Next Steps

It will be interesting to study the causality of the gap size between $F_{\mathrm{RC}}^{\mathrm{actual}}$ and $F_{\mathrm{CB}}^{\mathrm{predicted}}$ more closely. Additionally, it would be important to study the efectiveness of circuit benchmarking under a wider variety of noise models, and also a mix of them.

Acknowledgments

I would like to thank my supervisor Joseph Emerson for giving me the opportunity to work on this project, and for letting me present the theory behind circuit benchmarking at the American Physical Society (APS) Global Summit 2025 Conference in Anaheim, California.

I was solely responsible for all the coding, generated data sets, and plots for this project.

Chapter 3

PHYS 437A - Introduction

3.1 Background

To fully grasp the content of the project, a solid background in quantum information is needed. Nevertheless, a set of important definitions and concepts, which are frequently referenced throughout the project, is provided below.

Definition 3.1.1. [Easy Gates [5]] The easy gate set is the group generated by $C = \{\mathbf{P}_2, S\}$ defined as

$$\mathbf{C} = \{I, X, Y, Z, S, S^\dagger, SX, S^\dagger X\} \tag{3.1.1}$$

where $\mathbf{P}_2 = \{I, X, Y, Z\}$ is the one-qubit Pauli gate set

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{3.1.2}$$

and S is the phase flip gate defined as

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{3.1.3}$$

Definition 3.1.2 (Hard Gates [5])**.** The hard gate set is defined as

$$\mathbf{G} \ = \{H, CX, T\} \tag{3.1.4}$$

where H is the Hadamard gate

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.1.5}$$

CX is the CNOT gate

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.1.6}$$

and the T gate (also known as $\pi/8$ gate) is defined as

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \tag{3.1.7}$$

Remark: Easy and hard gates refer to gates with small and large amounts of noise [5].

Definition 3.1.3 (Pure/mixed state)**.** A pure state is a quantum state that can be represented by a vector $|\psi\rangle$ in a Hilbert space $\mathcal{H}$, with its density matrix given by $\rho = |\psi\rangle\langle\psi|$. A mixed state can only be described as a density matrix $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, where p_i are probabilities such that $\sum_i p_i = 1$.

Definition 3.1.4 (Coherent/Decoherent Error)**.** Coherent (unitary) errors transform pure states into other pure states and can be described by a unitary matrix operator on the Hilbert space. In contrast, decoherent (stochastic) errors, can transform pure states into mixed states, and thus cannot be represented as unitary operators.

Definition 3.1.5 (Markovian/Non-Markovian Error)**.** Markovian errors only depend on the current quantum state during the application of a quantum gate. However, non-Markovian errors may depend on the history of previous quantum states.

Definition 3.1.6 (Total Variation Distance (TVD) [6])**.** The TVD between two given probability distributions $\mathcal{P}, \mathcal{Q}$ is

$$d_{\mathrm{TV}}(\mathcal{P}, \mathcal{Q}) = \frac{1}{2} \sum_{x \in X} |\mathcal{P}(x) - \mathcal{Q}(x)| \tag{3.1.8}$$

Remark: The TVD quantifies the similarity between two probability distributions, where a TVD of 1 indicates complete overlap, and a TVD of 0 signifies no overlap.

Definition 3.1.7 (Pauli Channel [5])**.** The $n-$qubit Pauli channel is defined as

$$\mathcal{E}(\rho) = \sum_{P \in \mathcal{P}_n} p_P (P \rho P) \tag{3.1.9}$$

where $P \in \mathcal{P}_n$ represents a n-fold tensor product of Pauli matrices acting on the density matrix ρ, and satisfies

$$\sum_{P \in \mathcal{P}_n} p_P = 1, \quad p_P \geq 0 \tag{3.1.10}$$

Remark: The Pauli Channel is an example of a decoherent error model.

Definition 3.1.8 (Pauli Twirl [2])**.** The Pauli twirl is a process that modifies an arbitrary quantum channel into a decoherent Pauli channel by averaging the channel over the Pauli group as follows:

$$\mathcal{E}_{\mathrm{Twirled}}(\rho) = \frac{1}{|\mathcal{P}_n|} \sum_{P \in \mathcal{P}_n} P \mathcal{E}(P \rho P) P \tag{3.1.11}$$

Definition 3.1.9 (Average Gate Infidelity [2])**.**

$$r(\mathcal{E}_U, U) = 1 - \mathcal{F}(\mathcal{E}_U, U) = 1 - \int d\psi \, \langle \psi | U^\dagger \mathcal{E}_U(|\psi\rangle\langle\psi| \, U) | \psi \rangle \tag{3.1.12}$$

where $\mathcal{F}$ is the average gate fidelity, U is the ideal unitary map and $\mathcal{E}_U$ its noisy equivalent map. The average gate fidelity tells us how close the a noisy map is from its ideal map averaging over all the possible input states $|\psi\rangle$.

Remark: For simplicity, we will refer to $(\mathcal{E}_U, U)$ as $(\mathcal{E})$.

Definition 3.1.10 (Process Fidelity [2])**.**

$$F_P(\mathcal{E}) = \frac{\mathcal{F}(\mathcal{E})(d+1) - 1}{d} \tag{3.1.13}$$

where $d = 2^n$ for n qubits.

Definition 3.1.11 (Process Infidelity [2])**.**

$$e_F(\mathcal{E}) = r(\mathcal{E})\frac{d+1}{d} \tag{3.1.14}$$

Remark: We can see that the process infidelity contains the same information as the average gate infidelity, up to a dimensional constant.

3.2 Randomized Compiling

Randomized Compiling (RC) is a quantum error suppression protocol proposed by Joseph Emerson and Joel Wallman in 2015 [5]. RC transforms coherent and non-Markovian error channels into decoherent error channels, such as the Pauli channel, while preserving the logical circuit and maintaining the quantum circuit's depth. [5]. A decoherent error channel increases the likelihood of obtaining the correct output, as such channels exhibit a lower worst-case error rate compared to coherent error channels [6].

To begin, define a primitive gate set capable of generating universal quantum circuits. Any quantum circuit can be expressed as products and tensor products of gates in a universal gate set. This set can be further divided into two subsets: easy gates and hard

gates. The easy gate set has already been defined in Definition 3.1.1, while the hard gate set is described in Definition 3.1.2, and they indeed form a universal set of quantum gates; therefore, we can rewrite any circuit in terms of easy and hard gates. With these definitions in place, RC works as follows:

1. Define the desired quantum circuit.
2. Rewrite the circuit as alternating cycles of easy and hard gates.
3. Add random gates from the one-qubit Pauli set around the hard gate cycles, such that the resultant circuit is logically equivalent.
4. Compile the easy gates with the Pauli gates.
5. Run the compiled circuits and measure the results.
6. Repeat Steps 3-4-5 for multiple randomizations.
7. Add the measurement results from all randomly circuits and normalize them.
8. Calculate the total variation distance (TVD) of the normalized distribution in Step 7 with respect to the ideal distribution.

The following figures provide a visual representation of steps 2-3-4:

The circuits in steps 2 and 4 are equivalent; they can be represented by the same unitary matrix (up to a global phase). Repeating this process for multiple randomizations and averaging these randomly compiled circuits has a similar effect to applying a Pauli twirl (3.1.8) to the noisy channel, without altering the original circuit. As the number of randomizations increases, the effect approaches the exact effect of a Pauli twirl. Consequently, the noisy channel is transformed into an approximate decoherent Pauli channel. Decoherent channels are more likely to yield the correct output due to their lower worst-case error rate compared to coherent channels [6], making RC effective in reducing the total variation distance.

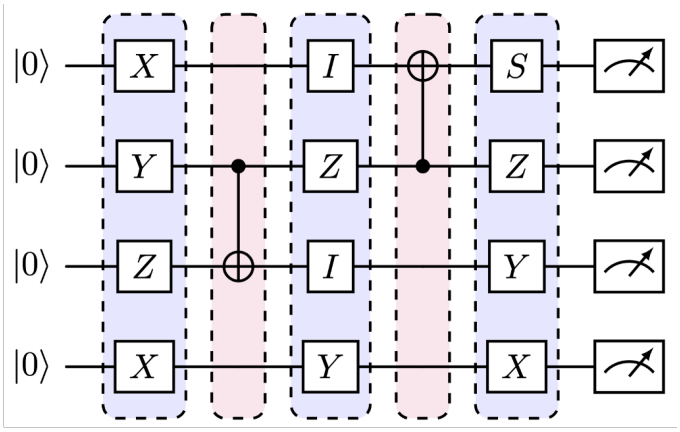


Figure 3.1: Step 2: Circuit written in alternating rounds of easy and hard cycles

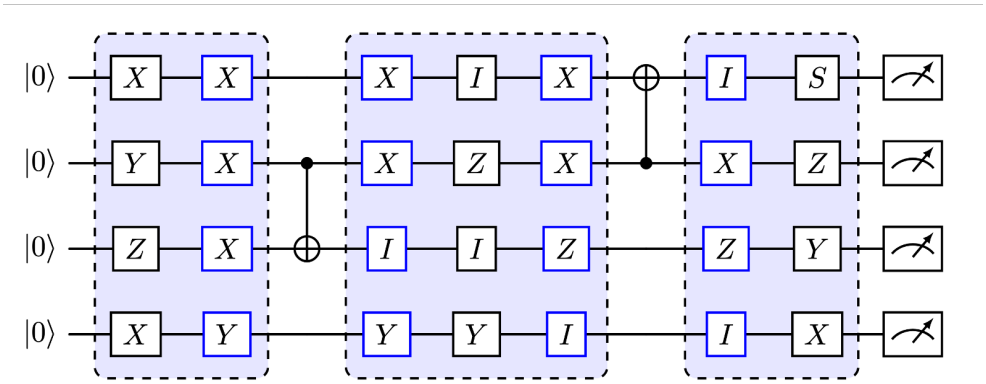


Figure 3.2: Step 3: Adding random Paulis around the hard gates

It is worth noting that generating the randomly compiled circuits requires classical computational resources. However, implementing RC needs additional quantum resources, as measurements must be taken from the randomly compiled circuits.

3.3 Circuit Benchmarking

Definition 3.3.1 (Circuit Benchmarking [8])**.** Let $\mathbf{C}$ be a circuit and n cycles and let $C = \{C_i\}_{i=1}^{n}$ be the set of the cycles of $\mathbf{C}$. Then,

$$F_P(\mathbf{C}_{RC}) \approx \prod_{i=1}^{n} F_{P_{CB}}(C_i) \tag{3.3.1}$$

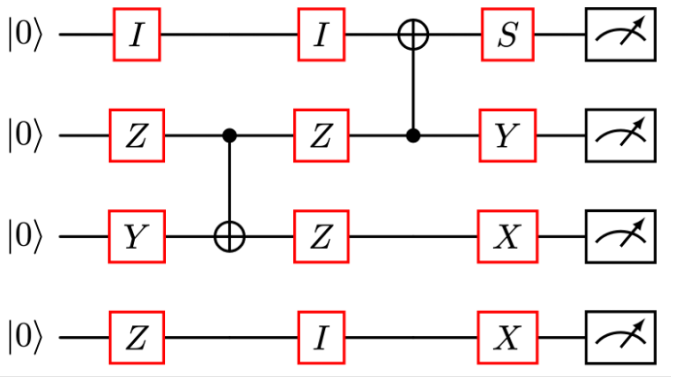


Figure 3.3: Step 4: Compiling the added Pauli gates with the easy gates

where $F_P(\mathbf{C}_{RC})$ represents the process fidelity of $\mathbf{C}$ under Randomized Compiling, and $F_{P_{CB}}(C_i)$ represents the process fidelity of the cycle C_i found under the Cycle Benchmarking. protocol.

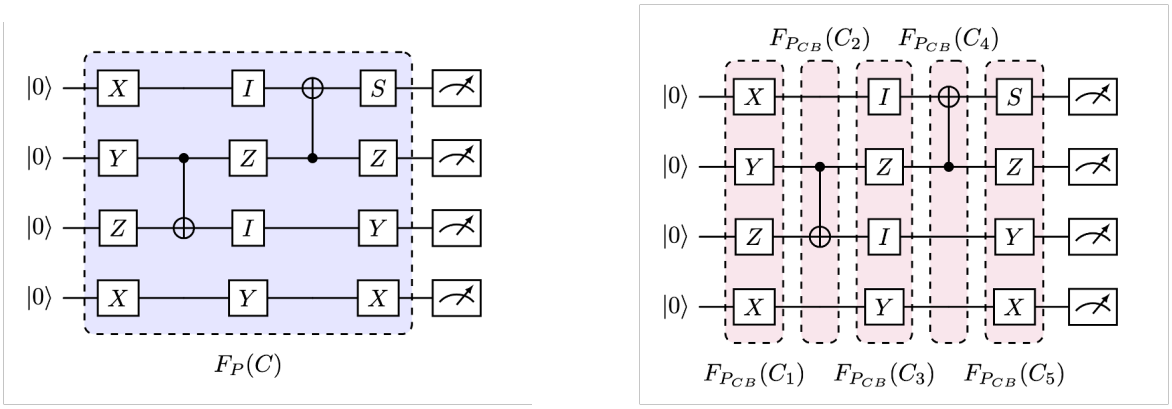


Figure 3.4: Visual representation of circuit benchmarking

Figure 3.4 illustrates the concept of circuit benchmarking, by decomposing the circuit into individual cycles to compute the process fidelity of the entire circuit. For reference, cycle benchmarking [4] is a fully scalable protocol used to determine the process fidelity of a quantum gate cycle of interest. Therefore, circuit benchmarking promises to be a scalable and efficient method to predict the process fidelity of a circuit under RC.

3.4 Connection Between Randomized Compiling and Circuit Benchmarking

Theorem 3.4.1 (TVD Bound [6]). Let $\mathcal{E}_{\text{coh}}$ be a noisy coherent channel, then

$$d_{\text{TV}}(\mathcal{P}_{\text{coh}}, \mathcal{P}_{\text{Ideal}}) \leq \sqrt{r(\mathcal{E}_{\text{coh}})}\sqrt{d(d+1)} \tag{3.4.1}$$

where $\mathcal{P}_{\text{coh}}, \mathcal{P}_{\text{Ideal}}$ represent the noisy and ideal probability distributions respectively. However, for any stochastic (decoherent) Pauli channel $\mathcal{E}_{\text{decoh}}$,

$$d_{\text{TV}}(\mathcal{P}_{\text{decoh}}, \mathcal{P}_{\text{Ideal}}) \leq r(\mathcal{E}_{\text{decoh}})\frac{d+1}{d} \tag{3.4.2}$$

Since $r(\mathcal{E}) \leq 1$, by inspection

$$r(\mathcal{E})\frac{d+1}{d} \leq \sqrt{r(\mathcal{E})}\sqrt{d(d+1)} \tag{3.4.3}$$

As a result, the TVD of a Pauli channel with respect to its ideal distribution has a tighter upper bound compared to that of a coherent channel. By tailoring coherent noise into a Pauli channel, RC effectively reduces the worst-case error rates. Therefore, for a sufficient number of randomizations, Equation 3.4.2 becomes

$$d_{\text{TV}}(\mathcal{P}_{\text{RC}}, \mathcal{P}_{\text{Ideal}}) \leq r(\mathcal{E}_{\text{RC}})\frac{d+1}{d} \tag{3.4.4}$$

where $\mathcal{P}_{\text{RC}}$ is the noisy probability distribution under RC and $r(\mathcal{E}_{\text{RC}})$ is the average gate infidelity of the channel $\mathcal{E}$ under RC. From Equation 3.4.4, we have

$$
\begin{aligned}
d_{\mathrm{TV}}(\mathcal{P}_{\mathrm{RC}}, \mathcal{P}_{\mathrm{ideal}}) &\leq r(\mathcal{E}_{RC}) \frac{d+1}{d} \\
&= e_F(\mathcal{E}_{RC}) \\
&= 1 - F_P(\mathcal{E}_{RC}) \\
&\approx 1 - \prod_i^n F_{P_{CB}}(C_i)
\end{aligned}
\tag{3.4.5}
$$

Therefore, we can express the TVD upper bound of a channel under RC as a function of the process fidelity. Moreover, we can approximate the process fidelity with circuit benchmarking. Therefore, we can use the results from circuit benchmarking to estimate the TVD upper bound under randomized compiling.

Chapter 4

PHYS 437A - Experiments and Results

4.1 Introduction

All experiments in this project were performed in the TrueQ framework [7]. TrueQ is a Python library which allows to run circuits on a simulator with different noise models, and apply protocols such as randomized compiling and cycle benchmarking. The noise model used for this project is an overrotation error model.

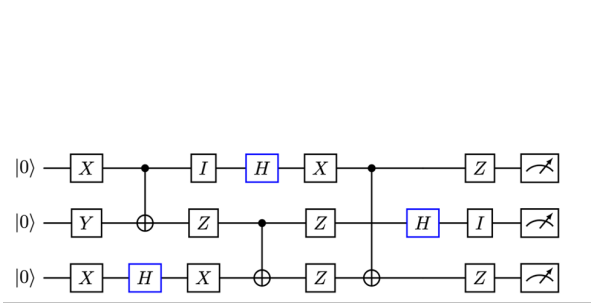
Definition 4.1.1 (Overrotation Error)**.** Let U be a unitary operator. An overrotation error occurs when, instead of applying U, the operation $U' = U^{1+\epsilon}$ is applied, where $\epsilon > 0$ represents the magnitude of the overrotation.

Remark: The overrotation error is an example of a coherent error, as it maps pure states into pure states.

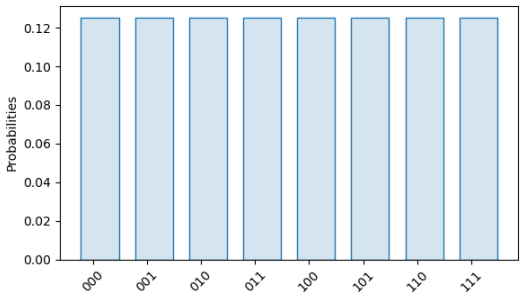
4.2 Randomized Compiling

4.2.1 Different Entropy Circuits under RC

We will define the **entropy of a circuit** as the number of states in superposition it produces in the computational basis. Recall that the Hadamard gate H maps the state $|0\rangle$ to the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. As an example, consider the four following circuits.

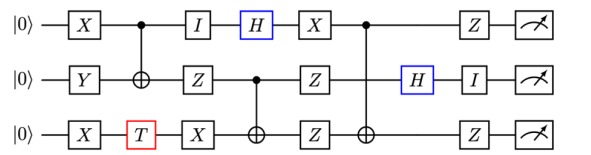


(a) Circuit

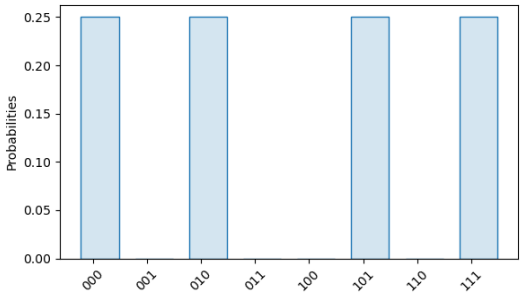


(b) Probability Distribution

Figure 4.1: Circuit with three Hadamard gates and its probability distribution



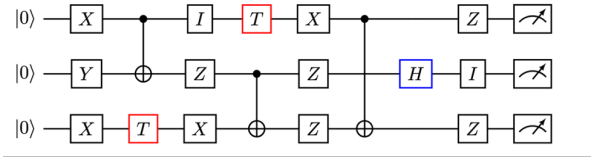
(a) Circuit



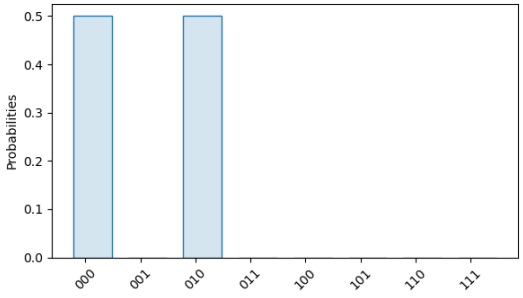
(b) Probability Distribution

Figure 4.2: Circuit with two Hadamard gates and its probability distribution

The circuits in Figures 4.1, 4.2, 4.3 and 4.4 share the same arrangement of easy and hard gate cycles. However, some Hadamard gates have been replaced with T gates (which

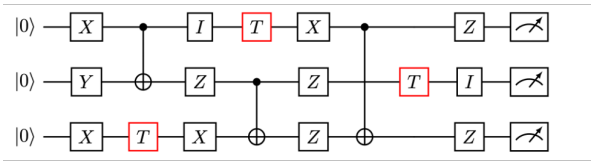


(a) Circuit

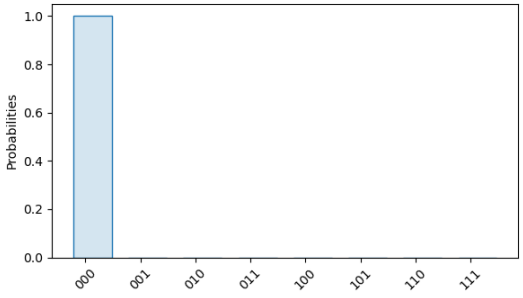


(b) Probability Distribution

Figure 4.3: Circuit with one Hadamard gates and its probability distribution



(a) Circuit



(b) Probability Distribution

Figure 4.4: Circuit with no Hadamard gates and its probability distribution

are also classified as hard gates). As expected, a Hadamard gate applied on a qubit doubles the number of states in superposition, thus increasing the circuit entropy. The circuit in Figure 4.1 is an example of a highest-entropy circuit, whereas the circuit in Figure 4.4 can be considered as a lowest-entropy cycle. According to [6], RC works best for high-entropy circuits. We will test this using TrueQ.

We will use the TrueQ simulator to run our circuits. The error model used is an overrotation error of magnitude of 0.1, which has only been applied to the hard gates; that is, the easy gates don't have noise. We will apply the RC protocol to our circuits using TrueQ's `randomly_compile` function, where we can input any circuit, along with a number of randomizations, n, and returns a list of n randomly compiled circuits.

For this experiment, we simulated 5-qubit circuits, that produced 2, 4, 8, 16, and 32 states in superposition. For each number of superposition states, we simulated circuits with 10, 15, 20, 25, 30, and 35 hard gate cycles. Additionally, for each configuration, 100 randomly generated circuits were evaluated. Finally, we applied the RC protocol for each randomly generated circuit with 20 randomizations. Each generated circuit includes a pair of CNOT gates and a single Hadamard gate applied to randomly selected qubits in each hard gate cycle.

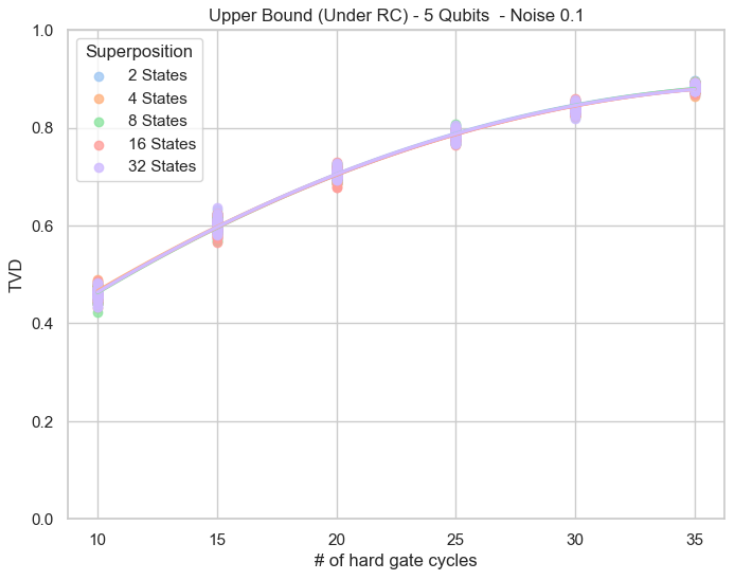


Figure 4.5: TVD upper bound under RC for 5-qubit circuits with different entropy

Figure 4.5 represents the upper bound defined in Equation 3.4.4 as a function of number of hard gate cycles. As we can see, the upper bound appears to remain nearly constant for a fixed number of hard gate cycles, regardless of the entropy of the circuit, and increases with the number of hard gate cycles.

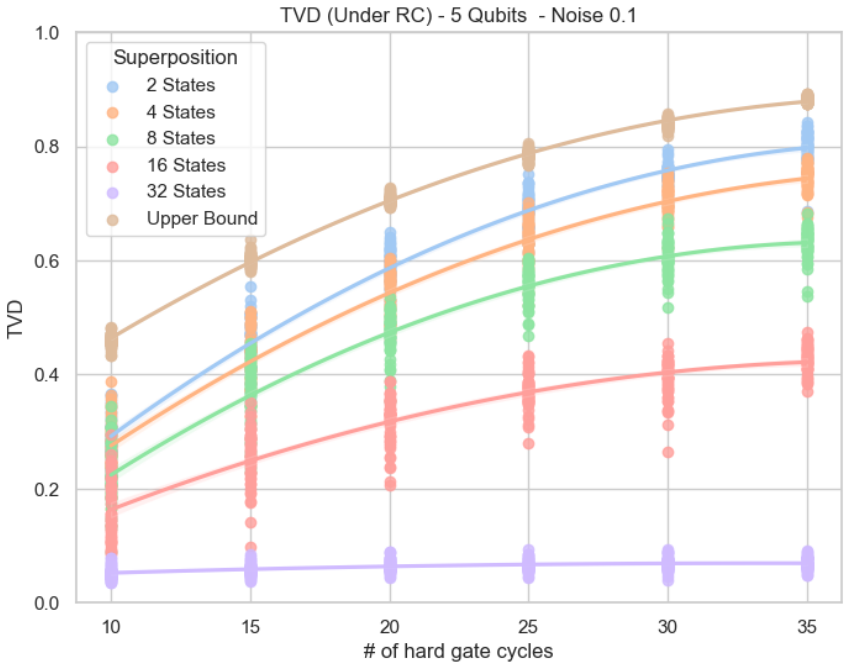


Figure 4.6: TVD under RC for 5-qubit circuits with different entropy

In contrast, Figure 4.6 reveals a clear correlation between the circuit's entropy and the TVD under RC; specifically, higher entropy corresponds to a lower TVD under RC. For circuits with maximum entropy (purple) the TVD remained close to 0, whereas circuits with the lowest-entropy (blue) had TVD values closest to the upper bound.

4.2.2 TVD vs Randomizations

In this section, we examine how the RC protocol reduces the TVD as the number of randomizations increases. Our analysis focuses on circuits with different entropies. Similar to Section 4.2.1, every hard gate cycle circuit consists of a pair of CNOT gates and one Hadamard gate, applied to randomly selected qubits.

For this part of the experiment, we apply RC to 5-qubit circuits with varying entropy levels, containing 20 hard gate cycles subject to an overrotation error of 0.1, while the number of randomizations is incremented from 1 to 40.

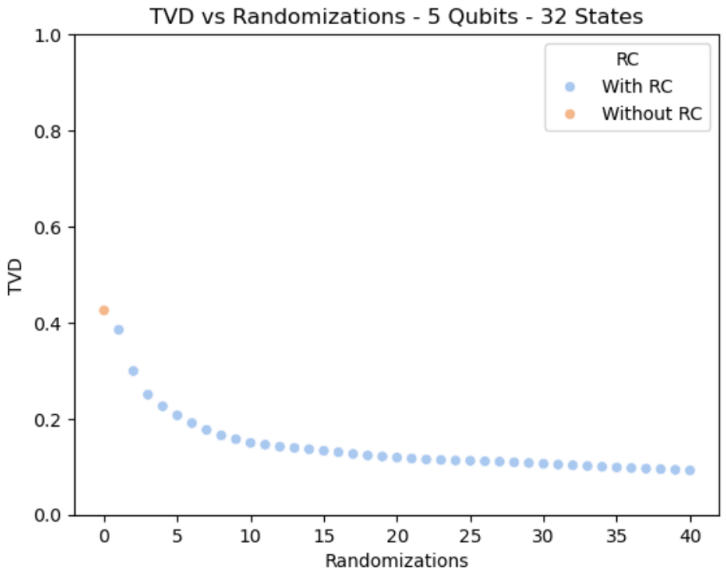


Figure 4.7: TVD vs Randomizations for circuit that produces 32 states in superposition

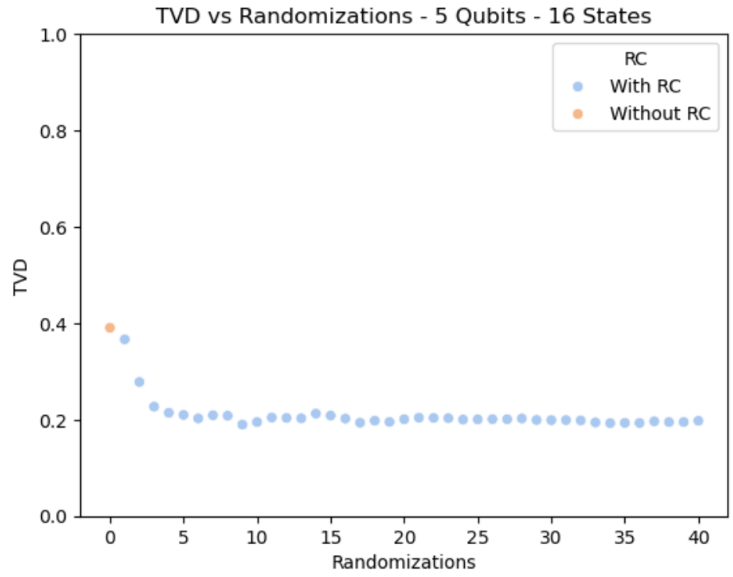


Figure 4.8: TVD vs Randomizations for circuit that produces 16 states in superposition

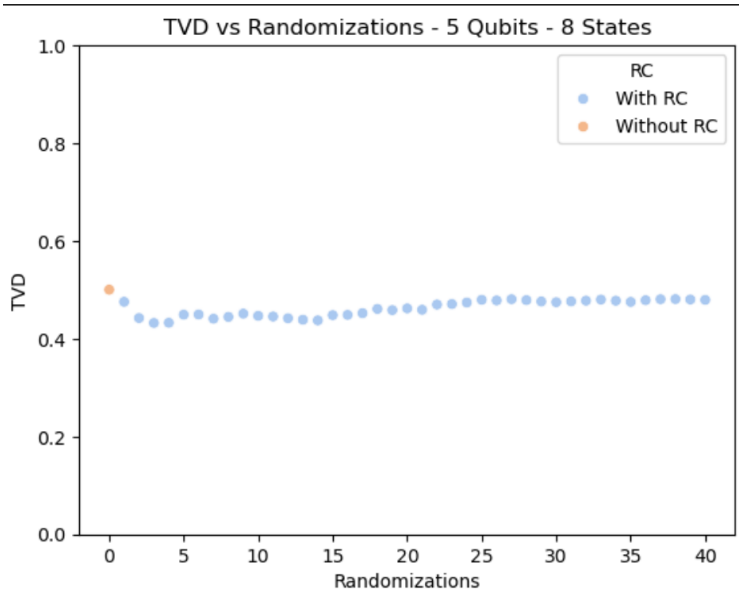


Figure 4.9: TVD vs Randomizations for circuit that produces 8 states in superposition

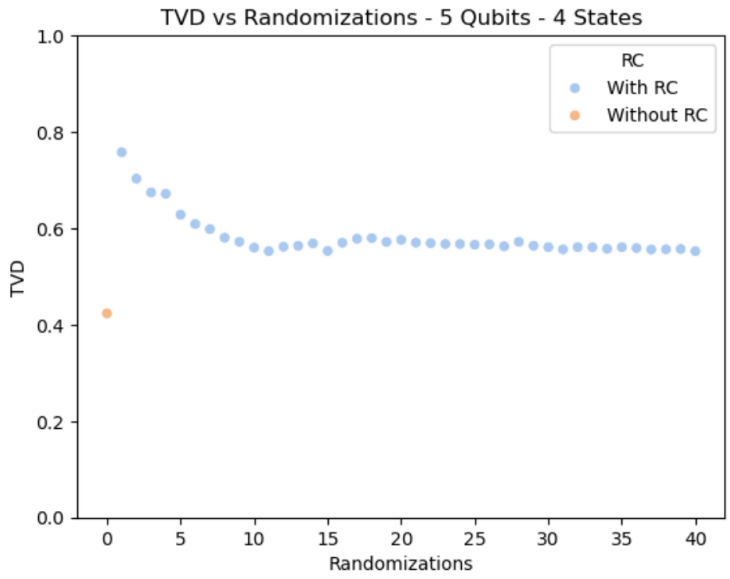


Figure 4.10: TVD vs Randomizations for circuit that produces 4 states in superposition

Each figure depicts a single randomly generated circuit with different entropy. The orange dot represents the TVD of the system without RC, whereas the blue dots correspond to the TVD values of the circuit under RC across different numbers of randomization. Figure 4.7 represents the case of applying RC to a highest-entropy 5-qubit circuit, which produces a superposition of 32 states (maximum entropy); the TVD decreases smoothly as the number of randomizations increases. However, as we reduce the number of states in superposition, the effectiveness of RC in reducing TVD diminishes. In Figure 4.8, the number of superpositions states is reduced in half, and the TVD doesn't decline after the first three randomizations. Figure 4.9 shows no evident decline in the TVD for a circuit that produces a superposition of 8 states, and lastly, Figure 4.10 shows an increase in the TVD, indicating a worsening effect for 4 superposition states. Therefore, the entropy of a circuit not only influences the reduction of TVD, but it can also increase it for low-entropy circuits. The reason why the circuit entropy affects the TVD performance is the following: If the final quantum state after executing the circuit is an eigenstate of the measurement basis, coherent noise does not influence the resulting probability distribution [6]. Since the primary goal of RC is to mitigate coherent noise, it becomes ineffective for low-entropy

circuits, as these circuits produce a superposition of a small number of eigenstates.

4.2.3 TVD with/without RC for Highest and Lowest Entropy Circuits

Sections 4.2.1 and 4.2.2 demonstrate that RC is most effective for high-entropy circuits, while its effectiveness decreases for low-entropy circuits. In this section we will compare the TVD before and after applying RC for the case of the highest-entropy and lowest-entropy circuit. The highest-entropy circuits have a pair of CNOT gates and one Hadamard gate in each hard gate cycle, while the lowest-entropy circuits include a pair of CNOT gates and a single T gate. The Hadamard gate is replaced with a T gate in the lowest-entropy circuits as T gates do not create superpositions of states, since it is extremely difficult to find lowest-entropy circuits containing Hadamard gates.

We will evaluate the TVD of randomly generated circuits with 10, 15, 20, 25, and 30 hard gate cycles with an overrotation error of magnitude of 0.1. For each number of hard gate cycles, we will produce 200 circuits. We will apply RC to each randomly generated circuit with 20 randomizations.

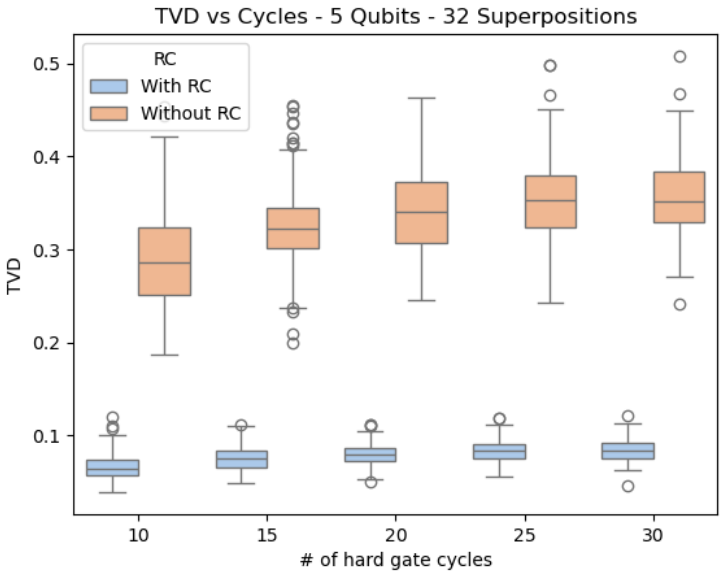


Figure 4.11: TVD with/without RC for highest-entropy circuits

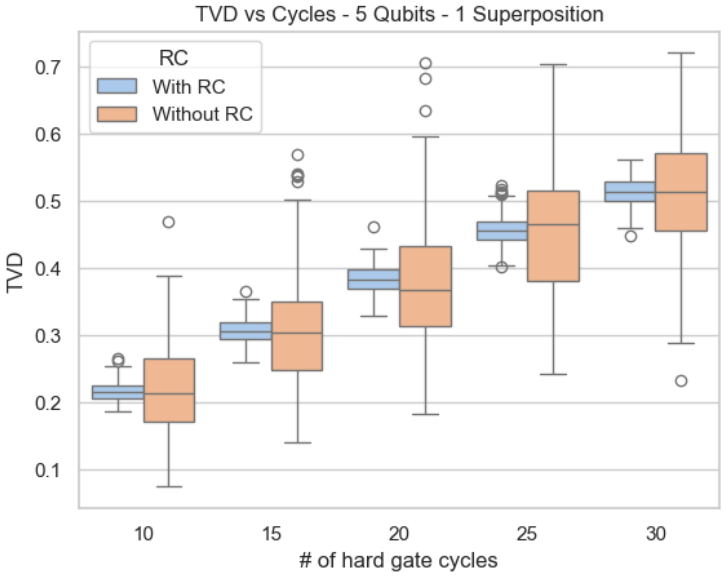


Figure 4.12: TVD with/without RC for lowest-entropy circuits

The boxplots in Figure 4.11 represent the TVD as a function of hard gate cycles with and without RC for highest-entropy circuits. As expected, the TVD under RC (blue) is extremely low compared to the TVD without RC (orange), confirming our results from Sections 4.2.1 and 4.2.2. We can observe, that as the number of cycles increases, both the TVD with and without RC slightly increase.

Figure 4.12 shows us an interesting result. The TVD under RC has a substantially smaller variance than the TVD without RC, yet their medians are extremely close. This suggests that for lowest-entropy circuits, RC may either increase or decrease the TVD. The boxplot reveals that RC effectively reduces the TVD in scenarios where the TVD without RC is exceptionally high; conversely, it tends to increase the TVD in cases where the original TVD is relatively low, thus reducing the variance.

4.3 Circuit Benchmarking

4.3.1 Testing Circuit Benchmarking

We will numerically illustrate Equation 3.3.1 for the case of 5-qubit circuits with an over-rotation error of magnitude of 0.1 in 2, 4, 6, and 8 hard gate cycles. For each number of hard gate cycles, 200 circuits were randomly generated. As shown in Figure 4.5, the process fidelity will not be affected by the entropy of the circuit; therefore, circuit entropy will not be a factor for the random generation of circuits for this section.

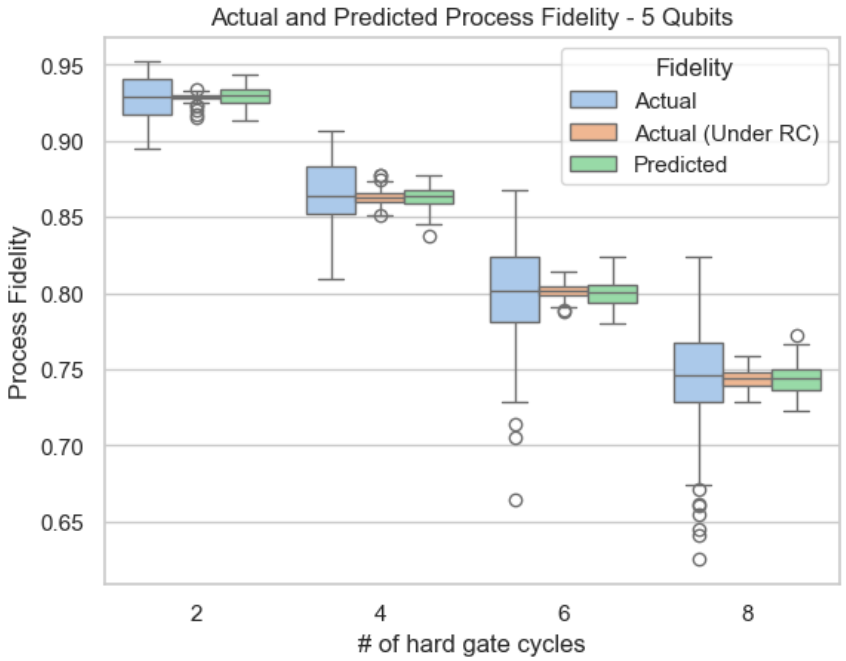


Figure 4.13: Actual Fidelity (with/without RC) and Predicted Fidelity

We can observe in Figure 4.13 the relationship between process fidelity and the number of hard gate cycles. We can see that all the process fidelities decrease as the number of cycles increases. The process fidelity for circuits without RC (blue) shows a high variance, which increases with the number of cycles. The actual fidelity under RC (orange) with 20 randomizations has tighter distributions, but its median is nearly identical to the actual fidelity without RC. Lastly, the predicted fidelity (green) consistently aligns with the actual

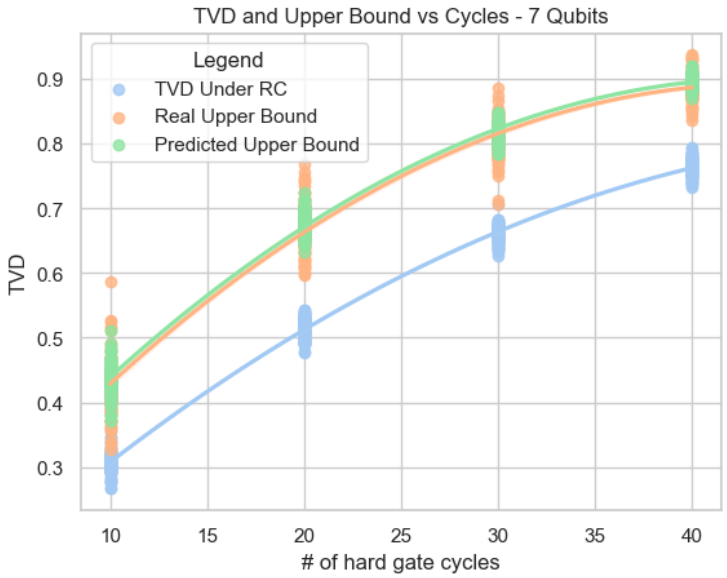


Figure 4.14: TVD and Upper Bound (Actual and Predicted) vs Cycles for a 7-qubit circuit

fidelity under RC, suggesting that the approximation shown in Equation 3.3.1 is true. This result suggests that we can approximate the upper bound of the TVD under RC using the predicted fidelity, as in Equation 3.4.5. Figure 4.14 shows the TVDs under RC with 20 randomizations (blue) alongside their actual (orange) and predicted (green) upper bounds of lowest-entropy 7-qubit circuits, with an overrotation error of magnitude of 0.1 on the hard gates, as a function of cycles. For each number of cycles, 200 circuits were randomly generated. Both the upper bound and TVD under RC increase with the number of hard gate cycles. As confirmed by Figure 4.13, the predicted upper bound closely aligns with the actual upper bound.

4.3.2 Adding Noise to the Easy Gates

We will show how adding noise to the easy gates affects the predicted fidelity. We will use 5-qubit randomly generated circuits with an overrotation error of magnitude of 0.1 in the hard gate and magnitude of 0.05 in the easy gates.

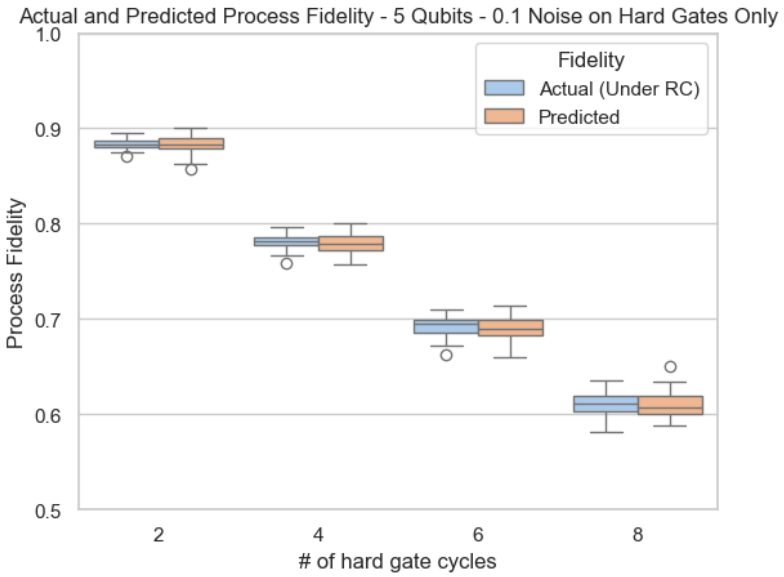


Figure 4.15: Actual and Predicted Fidelity for 5-qubit Circuits with Noiseless Easy Gates

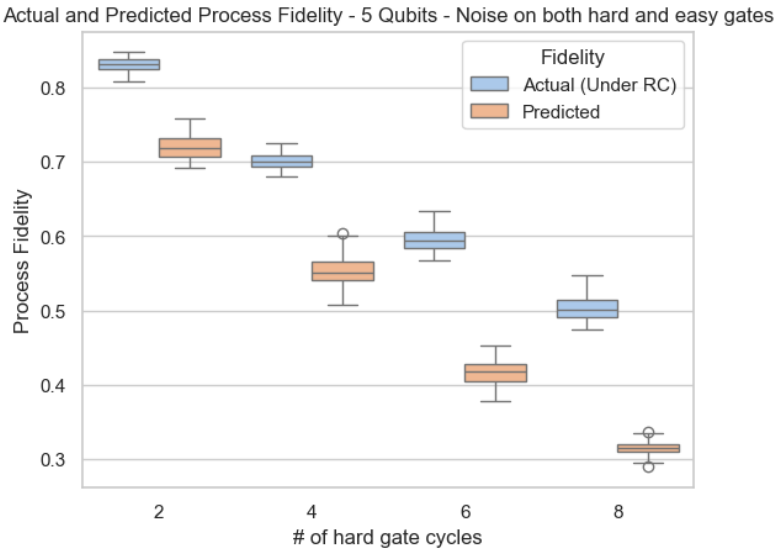


Figure 4.16: Actual and Predicted Fidelity for 5-qubit Circuits with Noisy Easy Gates

Figure 4.15 confirms the observation from Figure 4.13: when the easy gates are noiseless, the actual fidelity (blue) and the predicted fidelity (orange) are approximately equal. However, introducing the overrotation error to the easy gates causes a decrease in the predicted fidelity, as shown in Figure 4.16. Thus, Definition 3.3.1 is not applicable when the easy gates are subject to noise.

PHYS 437A - Concluding Remarks

From this project, the following conclusions can be drawn:

1. Randomized Compiling
    - Circuit entropy significantly influences the effectiveness of RC.
    - For high-entropy circuits, RC drastically reduces the TVD.
    - For low-entropy circuits, RC isn't effective, and it may even increase the TVD.
    - For high-entropy circuits, a fixed number of randomizations is sufficient for RC to be effective, regardless of the number of hard gate cycles.
2. Circuit Benchmarking
    - Equation 3.3.1 was numerically confirmed for 5-qubit circuits with overrotation errors in the hard gates.
    - Circuit entropy does not affect the process fidelity.
    - The predicted fidelity can be effectively used to estimate the upper bound of the TVD of a circuit under RC when the easy gates are noiseless.
    - When noise is added to easy gates, the predicted fidelity diverges from the actual fidelity, indicating that Equation 3.3.1 does not apply under such conditions.

PHYS 437A - Next Steps

It would be interesting to replicate the experiments from this project with different parameters, such as a larger number of qubits, and different noise models, such as the depolarizing error and relaxation error, and see how our results change. Additionally, we could continue using the overrotation error model but explore the impact of varying its magnitude. For PHYS 437 B next term, I will solely focus on circuit benchmarking, as there aren't any publications demonstrating its effectiveness yet, whereas randomized compiling has been more extensively studied. I would love to investigate the conditions required to ensure that Equation 3.3.1 holds true.

PHYS 437A - Acknowledgments

I would like to thank my supervisor, Joseph Emerson, for his extensive help and guidance during this semester and for providing me access to the TrueQ framework. I would also like to thank Virginia Frey and Arnaud Carignan-Dugas for identifying errors in my code that I would not have found by myself, and Debankan Sannamoth for helping me understand randomized compiling and cycle benchmarking.

I was solely responsible for all the coding, generated data sets, and plots for this project.

References

[1] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, January 1996. URL http://arxiv.org/abs/quant-ph/9508027. arXiv:quant-ph/9508027.

[2] Joseph Emerson. *Theory of Quantum Systems: From Mathematical Foundations to Experimental Methods with Applications to Quantum Computing*. 2021.

[3] Arnaud Carignan-Dugas, Joel Wallman, and Joseph Emerson. Validating quantum computation via circuit benchmarking (in progress).

[4] 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(1):5347, November 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-13068-7. URL https://www.nature.com/articles/s41467-019-13068-7. Publisher: Nature Publishing Group.

[5] Joel J. Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling, June 2016. URL http://arxiv.org/abs/1512.01098. arXiv:1512.01098.

[6] Akel Hashim, Ravi K. Naik, Alexis Morvan, Jean-Loup Ville, Bradley Mitchell, John Mark Kreikebaum, Marc Davis, Ethan Smith, Costin Iancu, Kevin P. O'Brien,

Ian Hincks, Joel J. Wallman, Joseph Emerson, and Irfan Siddiqi. Randomized compiling for scalable quantum computing on a noisy superconducting quantum processor, May 2021. URL http://arxiv.org/abs/2010.00215. arXiv:2010.00215.

[7] Stefanie J. Beale, Kristine Boone, Arnaud Carignan-Dugas, Anthony Chytros, Dar Dahlen, Hillary Dawkins, Joseph Emerson, Samuele Ferracin, Virginia Frey, Ian Hincks, David Hufnagel, Pavithran Iyer, Aditya Jain, Jason Kolbush, Egor Ospadov, José Luis Pino, Hammam Qassim, Jordan Saunders, Joshua Skanes-Norman, Andrew Stasiuk, Joel J. Wallman, Adam Winick, and Emily Wright. True-Q, June 2020. URL https://zenodo.org/records/3945250.

[8] Arnaud Carignan-Dugas, Matthew Alexander, and Joseph Emerson. A polar decomposition for quantum channels (with applications to bounding error propagation in quantum circuits), July 2019. URL http://arxiv.org/abs/1904.08897. arXiv:1904.08897.