IEEE Transactions on Quantum Engineering

Received 18 July 2023; accepted 9 August 2023; date of publication 15 August 2023;
date of current version 7 September 2023.

Digital Object Identifier 10.1109/TQE.2023.3305232

Testing Platform-Independent Quantum Error Mitigation on Noisy Quantum Computers

VINCENT RUSSO[1], ANDREA MARI[1], NATHAN SHAMMAH[1], RYAN LAROSE[1,2], AND WILLIAM J. ZENG[1,3]

[1]Unitary Fund, San Francisco, CA 94104 USA
[2]Institute of Physics, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland
[3]Goldman Sachs & Co., New York, NY 10004 USA

Corresponding author: Vincent Russo (e-mail: vincent@unitary.fund).

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing under Award DE-SC0020266 and Award DE-SC0020316 and in part by IBM under Sponsored Research Agreement W1975810.

ABSTRACT We apply quantum error mitigation (QEM) techniques to a variety of benchmark problems and quantum computers to evaluate the performance of QEM in practice. To do so, we define an empirically motivated, resource-normalized metric of the improvement of error mitigation, which we call the improvement factor, and calculate this metric for each experiment we perform. The experiments we perform consist of zero-noise extrapolation and probabilistic error cancellation applied to two benchmark problems run on IBM, IonQ, and Rigetti quantum computers, as well as noisy quantum computer simulators. Our results show that error mitigation is, on average, more beneficial than no error mitigation—even when normalized by the additional resources used—but also emphasize that the performance of QEM depends on the underlying computer.

INDEX TERMS Quantum computing.

I. INTRODUCTION

Quantum computers have steadily improved over the past two decades, as can be seen in component metrics like T1 and T2 times [20], [46] as well as full system metrics like quantum volume [8]. While we expect these hardware improvements to continue, it is generally accepted that error rates cannot be made low enough purely by hardware improvements. Rather, to achieve error rates low enough for useful applications, hardware improvements should be coupled with algorithmic or software methods.

The most commonly pursued algorithmic method is quantum error correction [13], [42], [48], which generally provides a tradeoff in qubit quantity for qubit quality, i.e., using more qubits to achieve a lower logical error rate. Today, the state-of-the-art experiments in quantum error correction [2], [3] confirm an exponential suppression of errors as the code distance increases, but do so for relatively small code distances. For example, the largest surface code implementation to our knowledge [3] uses 49 physical qubits to encode one logical qubit in a distance five surface code, while rough estimates for current error rates require around 1000 physical qubits per logical qubit for fault tolerance.

Because the experimental requirements of quantum error correction are very demanding, and because of widespread interest in applications of noisy quantum computers [34], a new set of algorithmic methods to deal with errors has emerged in recent years. These new methods are referred to as quantum error mitigation (QEM) [7], [16], and are designed to be less experimentally demanding than full quantum error correction. However, this comes at the cost of being less general and more heuristic than quantum error correction.

While a relatively large number of error mitigation techniques have been proposed [19], [23], [37], [39], [51], [55], [57], there have been relatively few experiments using error mitigation, despite the fact that error mitigation is specifically designed for current quantum computers. A summary from the literature of QEM experiments performed on quantum computers is shown in Table 1. Note that this table is not exhaustive for all benchmarks outside of the context of QEM.

TABLE 1 History of Quantum Error Mitigation Experiments on Quantum Computers in the Literature

QEM	Benchmark	Qubits n	Computer(s)	Ref.
ZNE	RB	1, 2	5-qubit superconducting device	[KTC+19]
	RB	2	IBMQ London & Rigetti Aspen-8	[LMK+22]
	RB, PG	2 - 5	IBMQ Lagos & IBMQ Casablanca	[CDM+22]
	RB, MC	3, 5, 12	IBMQ Lima, IBMQ Kolkata, Rigetti Aspen-M2, IonQ Harmony	(This work)
	VQE	4	5-qubit superconducting device	[KTC+19]
	QV	5	IBMQ Belem, IBMQ Lima, & IBMQ Quito	[LMR+22]
	RB, TE	26	27-qubit superconducting device	[KWY+21]
PEC	RB	2	2-qubit trapped ion ($^{171}Yb^+$) device	[ZLZ+20]
	RB, MC	3	Rigetti Aspen-M2, IBMQ Lima, IonQ Harmony	(This work)
	CYC	4	4-qubit superconducting device	[FHV+22]
	TE, CYC	4, 10	27-qubit superconducting device	[BMKT22]
DD	IDLE	1, 2	IBMQX4, IBMQX5, & Rigetti Acorn	[PAFL18]
	Adder, GHZ, QAOA, QFT, VQE	4, 5, 6	IBMQ Guadalupe, & IBMQ Jakarta	[SRM+22]
	QPE	5	IBMQ Paris, IBMQ Guadalupe, & IBMQ Toronto	[DTDQ21]
	QV	6	IBMQ Montreal	[JJAB+21]
	QFT	6, 7	IBMQ Paris, IBMQ Guadalupe, & IBMQ Toronto	[DTDQ21]
	BV	7, 8	IBMQ Paris, IBMQ Guadalupe, & IBMQ Toronto	[DTDQ21]
	QAOA	8, 10	IBMQ Paris, IBMQ Guadalupe, & IBMQ Toronto	[DTDQ21]
CDR	RB, PG	2 - 5	IBMQ Lagos & IBMQ Casablanca	[CDM+22]
	VQE	5	IBMQ Rome	[CACC21]
	VQE	6	IBMQ Toronto	[CMSC22]
	VQE	16	IBMQ Almaden	[ZCN+22]
SSE	VQE	2	2-qubit superconducting device	[CRD+18]
	VQE	2	3-qubit superconducting device	[SBMS+19]
VD	GHZ	5	5-qubit trapped ion, UMD, ($^{171}Yb^+$) device	[SCZ+22]

Quantum error mitigation (QEM) technique acronyms: ZNE = zero-noise extrapolation, PEC = probabilistic error cancellation, also referred to as quasi-probabilistic decomposition (QPD) by some authors, DD = dynamical decoupling, SSE = subspace expansion, VD = virtual distillation, CDR = clifford data regression. Benchmark acronyms: RB = randomized benchmarking, VQE = variational quantum Eigensolver, TE = time evolution, IDLE = allowing a state to idle (identity operation), CYC = alternate cycles of single-qubit layers and clifford layers, GHZ = Greenberger–Horne–Zeilinger, MC = mirror circuits, Adder = ripple carry adder, PG = pauli gates, QV = quantum volume, QAOA = quantum approximate optimization algorithm, QFT = quantum Fourier transform, QPE = quantum phase estimation, BV = Bernstein-Vazirani.

For instance, this review [44] provides a thorough collection of T1 and T2 times for the dynamical decoupling benchmark.

In this work, we evaluate QEM in practice using a suite of experiments on various benchmarks and quantum computers. We consider two error mitigation techniques, two benchmark problems, and four quantum computers. To quantify the performance of QEM (relative to no error mitigation), we define a natural metric that we call the *improvement factor*.

Our results show that QEM improves the performance of noisy quantum computations in nearly all experiments we consider, even when normalized by the additional resources (namely, samples) used in the error mitigation techniques. Depending on the number of qubits, circuit depth, and particular computer in the experiment, our results show between a $1\times$ and $7\times$ improvement from QEM. Further, the error mitigation we use is "out-of-the-box" in that it is not tailored to the benchmark problems or computers we consider. Because of this, we expect QEM to be an essential component of NISQ and even error-corrected computations and offer perspective on these points. The rest of this article is organized as follows. Section II describes our methods for assessing the performance of QEM in practice. This includes our definition of the improvement factor (Section II-A), the error mitigation techniques (Section II-B), the benchmark problems (Section II-C), and the quantum computers (Section II-D) used in our experiments. We present the results of our experiments in Section III, and we discuss them in the larger context of QEM and quantum computation in Section IV. Finally, Section V concludes this article.

II. METHODS

To assess the experimental performance of QEM, we define a natural measure comparing the accuracy of an experiment with QEM to the accuracy without QEM. This measure, which we call the *improvement factor*, is motivated and defined in Section II-A. We experimentally calculate this measure using two QEM techniques (Section II-B) with two benchmark problems (Section II-C) on four quantum computers and three noisy quantum computer simulators (Section II-D). All the error mitigation techniques for this study were implemented using the Mitiq error mitigating compiler [27].

A. IMPROVEMENT FACTOR

The goal of most QEM techniques is to improve the estimation of expectation values. Following an empirical approach, we quantify the improvement in error mitigation by comparing the estimation errors obtained with and without error mitigation.

Let ρ be an ideal n-qubit quantum state prepared by a noiseless quantum computer after the execution of some given quantum circuit C, i.e., $\rho = C|0^{\otimes n}\rangle\langle 0^{\otimes n}|C^\dagger$. For an

observable $\hat{A} = \hat{A}^\dagger$, the ideal (noiseless) expectation value is

$$A = \mathrm{tr}\left[\rho \hat{A}\right] = \mathrm{tr}\left[C|0^{\otimes n}\rangle\langle 0^{\otimes n}|C^\dagger \hat{A}\right]. \tag{1}$$

When using a noisy quantum computer, we instead prepare a noisy state ρ' and collect N shots (samples) to obtain an empirical estimate A' of the expectation value.

The goal of QEM is to compute some quantity A_{QEM} which is a more accurate estimate of the ideal expectation value A compared to the unmitigated estimate A'.[1] Generally, computing A_{QEM} is done by executing a set of circuits $\{C_1, \ldots, C_{k_{\mathrm{QEM}}}\}$ related to C—usually with a different number of qubits, gates, and/or total shots N_{QEM}—then postprocessing the noisy results to obtain the error-mitigated estimate A_{QEM}. We refer to an evaluation of an unmitigated expectation value A' as a trial. After performing t trials $A'_{[1]}, \ldots, A'_{[t]}$ we can quantify the estimation error through the root-mean-square error (rmse)

$$\sqrt{\frac{1}{t}\sum_{i=1}^{t}\left(A'_{[i]} - A\right)^2}. \tag{2}$$

We use the rmse since it reduces to the absolute error when all $A'_{[i]}$ are approximately equal (e.g., in the limit of large N) and, at the same time, it also takes into account the estimation error due to the statistical fluctuations of the results $A'_{[i]}$ over different trials.

Similarly, we refer to an evaluation of A_{QEM} as a QEM trial. After performing t QEM trials $A^{[1]}_{\mathrm{QEM}}, \ldots, A^{[t]}_{\mathrm{QEM}}$ we evaluate the rmse

$$\sqrt{\frac{1}{t}\sum_{i=1}^{t}\left(A^{[i]}_{\mathrm{QEM}} - A\right)^2}. \tag{3}$$

As noted, evaluating each $A^{[i]}_{\mathrm{QEM}}$ potentially uses additional resources in the form of circuits, qubits, gates, and/or shots. To account for these additional resources, we define the *problem-specific improvement factor*

$$\mu_{\mathrm{QEM}}\left(C, \hat{A}\right) := \frac{1}{c}\frac{\sqrt{\sum_{i=1}^{t}\left(A'_{[i]} - A\right)^2}}{\sqrt{\sum_{i=1}^{t}\left(A^{[i]}_{\mathrm{QEM}} - A\right)^2}} \tag{4}$$

i.e., the cost-normalized ratio of rmse, where c is a cost factor that takes into account the additional resources that error mitigation requires compared to a simple unmitigated experiment. One can define the cost factor c in many different ways depending on the specific context and practical constraints [43]. For example, a possible definition is

$$c = \sqrt{N_{\mathrm{QEM}}/N} \tag{5}$$

such that error mitigation strategies which require a large number of shots have $c > 1$ and, therefore, a reduced improvement factor (see Section IV-B for a discussion on normalizing by other resources, e.g. qubits, and gates, in addition to shots). For all the hardware experiments presented in this work, we use the same number of shots $N = N_{\mathrm{QEM}}$, such that $c = 1$. However, via numerical simulations, we also test the definition of the improvement factor for different values of c. In particular, in Section III-A, we compare the normalized and unnomralized improvement factor for $N \neq N_{\mathrm{QEM}}$.

The improvement factor μ_{QEM} is a natural, empirically defined measure of the performance of QEM for a specific expectation value problem defined by a circuit C and observable $\hat{A}$. To generalize over different problems in addition to averaging over multiple trials, we also average over a set of circuits $\mathcal{C}$ and a set of observables $\hat{\mathcal{A}}$ to define the *improvement factor*

$$\mu_{\mathrm{QEM}} := \frac{1}{c}\frac{\sqrt{\sum_{C\in\mathcal{C},\hat{A}\in\hat{\mathcal{A}}}\sum_{i=1}^{t}\left(A'_{[i]} - A\right)^2}}{\sqrt{\sum_{C\in\mathcal{C},\hat{A}\in\hat{\mathcal{A}}}\sum_{i=1}^{t}\left(A^{[i]}_{\mathrm{QEM}} - A\right)^2}}. \tag{6}$$

Here, as in (4), the circuit C is implicit in the expectation values A, $A'_{[i]}$, and $A^{[i]}_{\mathrm{QEM}}$, e.g., $A = \mathrm{tr}[C|0\rangle\langle 0|C^\dagger \hat{A}]$. While this definition is general with respect to the circuits C, experimentally we consider two classes of benchmark circuits (Section II-C) and quote the results from these classes of circuits separately. Indeed, for most experiments on quantum computers, we generate $|\mathcal{C}| = 4$ randomized instances of benchmark circuits from the two classes. We choose benchmark circuits such that there is one natural observable for each circuit, i.e., $|\hat{\mathcal{A}}| = 1$, where $|\cdot|$ denotes the cardinality of the set. In all cases, due to limited device availability, we perform $t = 1$ trial for each $C, \hat{A} \in \mathcal{C} \times \hat{\mathcal{A}}$. In the numerical simulation presented in Section III-C instead, we also explored the multitrial regime, up to $t = 30$.

We note that Cirstoiu et al. [9] also define a measure of the improvement from error mitigation, in particular a problem-specific measure. This quantity, which they call the relative mitigation error and denote by ϵ, is given by (in the notation of this article)

$$\epsilon_{\mathrm{QEM}}(C, \hat{A}) := \frac{|A_{\mathrm{QEM}} - A|}{|A' - A|}. \tag{7}$$

For $t = 1$, $\mu_{\mathrm{QEM}}(C, \hat{A}) = c\,\epsilon^{-1}_{\mathrm{QEM}}(C, \hat{A})$.

Finally, we also note that different error mitigation techniques can have different performances depending on the locality of A; the observable of interest. The number of local qubits in which A acts nontrivially, is often called the *weight*. In most of the experiments presented in this work, we consider maximum-weight observables corresponding to projectors on a given computational basis state, i.e., $A = |b\rangle\langle b|$. However, in Section III-B, we also analyze improvement factors associated with observables having different weights.

[1] Note that here and throughout, we use the acronym QEM to refer to a generic QEM technique and a specific acronym for a specific QEM technique. So, for example, the zero-noise extrapolated expectation value of $\hat{A}$ is denoted A_{ZNE}, and similarly for other quantities. A summary of our notation is included in Appendix A.

B. QUANTUM ERROR MITIGATION TECHNIQUES

1) ZERO-NOISE EXTRAPOLATION

We apply zero-noise extrapolation (ZNE) [23], [25], [50] with both linear and Richardson extrapolation—which we, respectively, denote ZNE(L) and ZNE(R)—to our benchmark problems. For both cases, we evaluate $k_{\text{ZNE}} = 3$ noisy expectation values $A'(\lambda_i)$ at different noise scale factors $\lambda_i \in \{1, 2, 3\}$ and the zero-noise limit is obtained as a linear combination of the results

$$A_{\text{ZNE}} = \sum_{i=1}^{k_{\text{ZNE}}} \eta_i A'(\lambda_i). \tag{8}$$

For Richardson extrapolation, the best fit coefficients η_i in (8) are given by the following [19]:

$$\eta_i := \prod_{j \neq i} \frac{\lambda_j}{\lambda_j - \lambda_i}. \tag{9}$$

For linear extrapolation, the coefficients η_i are obtained from a linear best fit and also only depend on the noise scale factors, but the analytical expression is more involved [19, eq. (26)].

For all ZNE experiments, we use global unitary folding [19] to scale noise. For odd integer scale factors λ_i, this amounts to replacing the circuit C by $C(C^\dagger C)^{(1-\lambda_i)/2}$. If λ_i is not an odd integer, a fraction of the full circuit is folded and appended to the circuit as described in [19]. Each noise-scaled circuit is executed with $\lfloor N/k_{\text{ZNE}} \rfloor = \lfloor 10^4/3 \rfloor$ shots so that $N_{\text{ZNE}} \simeq N = 10^4$ (i.e., so that we use the same total number of shots in ZNE as in the unmitigated experiment). For more details on our ZNE implementation, see Appendix B41.

2) PROBABILISTIC ERROR CANCELLATION

We also apply probabilistic error cancellation (PEC) [15], [50], [57] to each benchmark problem. Here, the first step is to characterize the set of noisy, implementable operations $\{\mathcal{O}_\alpha\}$ of a computer so that we can represent the ideal (noiseless) operations $\{\mathcal{G}_i\}$ of a circuit in this basis, namely

$$\mathcal{G}_i = \sum_\alpha \eta_{i,\alpha} \mathcal{O}_\alpha. \tag{10}$$

Note that the calligraphic symbols $\mathcal{G}_i$ and $\mathcal{O}_\alpha$ stand for super-operators acting on the quantum state of the qubits as linear quantum channels, and $\eta_{i,\alpha} \in \mathbb{R}$. In principle, this requires full tomographic knowledge of the noisy operations $\{\mathcal{O}_\alpha\}$, but we make two simplifying assumptions in our experiments.

1) We neglect errors of single-qubit gates.
2) We assume that all two-qubit gates $\mathcal{G}_{2Q}$ (CNOT or CZ in our experiments) are followed by local depolarizing noise, i.e.,

$$\mathcal{G}_{2\text{Q}}^{(\text{noisy})} = (\mathcal{D}_p \otimes \mathcal{D}_p) \circ \mathcal{G}_{2Q} \tag{11}$$

where $\mathcal{D}_p(\rho) = (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z)$ is the single-qubit depolarizing channel and p is the local error probability.

Under these assumptions, the quasi-probability representation of the ideal $\mathcal{G}_{2Q}$ gate can be derived for any value of p [49], [50]. For each $\mathcal{G}_{2Q}$ operation, we obtain p from the error rate in the calibration data reported by the hardware vendor for the associated two-qubit gate (IBM, Rigetti), or from the average two-qubit error rate (IonQ). More precisely, in (11), the overall two-qubit error probability is $p_{2Q} = 1 - (1-p)^2$. So, given the parameter p_{2Q} reported by the calibration data of the computer, we estimate $p = 1 - \sqrt{1 - p_{2Q}}$ and use this in (11) to obtain the basis $\mathcal{O}_\alpha$.

After obtaining the basis $\mathcal{O}_\alpha$, we represent all two-qubit gates G_i in the circuit in this basis as in (10) and stochastically sample $k_{\text{PEC}} = 100$ new circuits to execute. Each circuit is executed with $N/k_{\text{PEC}} = 10^4/100$ shots so that $N_{\text{PEC}} = N = 10^4$ (i.e., so that we use the same total number of shots in PEC as in the unmitigated experiment). For more details on our implementation of PEC, see Appendix B42.

C. BENCHMARK PROBLEMS

A benchmark problem is defined by an n-qubit, depth d quantum circuit C, and an observable $\hat{A}$ as in (1). In this work, we consider two benchmark problems in which the circuit C produces (without noise) a single bitstring $z_C \in \{0, 1\}^n$, and we always take $\hat{A} = |z_C\rangle\langle z_C|$ as the corresponding observable. Both circuits have a number of qubits n and a depth d which can be varied independently, and we use $|\mathcal{C}| = 4$ (random) instances of each circuit for a given n, d. In experiments on quantum computers, we choose $n \in \{3, 5\}$ and $d \subseteq \{1, 3, 5, 7, 9\}$. Additionally, for a specific device (IBMQ Kolkata), we also perform a larger experiment with $n = 12$ qubits and $d \in \{1, 5, 9\}$. The number of one- and two-qubit gates for a given n, d depends on the circuit type and is discussed for each circuit type below. As discussed in Section II-D5 we repeat each of these experiments on noisy quantum computer simulators for comparison.

1) RANDOMIZED BENCHMARKING

We use randomized benchmarking (RB) circuits [10], [18], [29], [30], [31] as one benchmark problem in our experiments. An n-qubit, depth d RB circuit is a sequence of d random Clifford group elements $U_d U_{d-1} \cdots U_1$, followed by a (classically computed) inverse element $U_{\text{inv}} = (U_d U_{d-1} \cdots U_1)^{-1}$, such that the full circuit

$$C = U_{\text{inv}} U_d U_{d-1} \cdots U_1 \tag{12}$$

is the identity operation (without noise). As such, the only bitstring that should be measured is $z_C = 0^n$, and we take the observable to be $\hat{A} = |z_C\rangle\langle z_C| = |0\rangle\langle 0|^{\otimes n}$.

In all experiments, we use a line of qubits and apply 2-qubit RB sequences to each neighboring pair of qubits on the line. If the total number of qubits is odd we also apply a 1-qubit RB sequence to the last qubit. The rationale for this

TABLE 2 Average Number of Two-Qubit (Single-Qubit) Gates for an n-Qubit, Depth d RB Circuit

d	$n = 3$	$n = 5$	$n = 12$
1	3 (22)	7 (39)	19 (99)
3	6 (44)	11 (73)	36 (204)
5	9 (64)	18 (118)	53 (307)
7	12 (80)	24 (150)	73 (403)
9	15 (108)	31 (194)	89 (506)
12	18 (135)	37 (242)	115 (651)

The average is taken over ten random instances. Note that the number of single-qubit gates may differ on different hardware due to the final compilation into native gates, but the number of two-qubit gates is hardware-independent.

TABLE 3 Average Number of Two-Qubit (Single-Qubit) Gates for an n-Qubit, Depth d Mirror Circuit

d	$n = 2$	$n = 5$	$n = 12$
1	2 (26)	4 (41)	10 (95)
3	6 (46)	12 (68)	30 (158)
5	10 (68)	20 (98)	51 (223)
7	14 (87)	28 (125)	72 (284)
9	18 (108)	36 (154)	92 (350)
12	24 (138)	48 (197)	121 (448)

The average is taken over ten random instances. Note that the number of single-qubit gates may differ on different hardware due to the final compilation into native gates, but the number of two-qubit gates is hardware-independent.

choice is that a linear topology can be easily embedded in the connectivity graph of all quantum computers we consider in this work, and therefore this choice makes our benchmarks more consistent across different computers.

Note that the parameter d is the number of random Clifford elements, not the actual physical depth of the circuit. Each Clifford element must be decomposed into two-qubit gates and single-qubit gates. The total number of two-qubit gates depends on both d and the number of qubits n. In Table 2, we report the average number of two-qubit and single-qubit gates used in our experiments.

2) MIRROR CIRCUITS

We also use mirror circuits [35], [36] as a benchmark problem. Mirror circuits are similar to RB circuits in that they have a structure of random layers, but their final state $|z_C\rangle$ is randomized. This configuration allows a more uniform sampling of measurement errors.

An n-qubit, depth d mirror circuit C is a randomized sequence of d Clifford layers and Pauli layers, where Clifford layers are organized in such a way to conjugate a Pauli layer into a rotated Pauli layer (see Fig. 1 of [36]). The full mirror circuit C is equivalent to a random Pauli operator $\mathcal{P}$, thus the final ideal (noiseless) state is a random computational basis state $|z_C\rangle = \mathcal{P}|00\ldots0\rangle$, and we take the observable to be $\hat{A} = |z_C\rangle\langle z_C|$.

It is important to notice that the Clifford depth d reported in our results for mirror circuits is the number of random Clifford layers. Since each Clifford layer is compiled into elementary gates, and since additional random Pauli layers are present in the circuit, the Clifford depth d is different from the number of physical gates applied in the circuit. The average number of two-qubit and single-qubit gates used in our experiments are reported in Table 3 for different values of d and n.

D. QUANTUM COMPUTERS

We test error mitigation techniques with each benchmark circuit on four quantum computers—IBMQ Kolkata, IBMQ Lima, Rigetti Aspen-M2, and IonQ Harmony—shown in Fig. 1 and described in the following sections. We also perform experiments on noisy quantum computer simulators for comparison to hardware and for additional experiments. The noise models we use are described in Section II-D5.

1) IBMQ LIMA

The IBMQ Lima computer consists of five superconducting transmon qubits arranged in a "T-shape" topology shown in Fig. 1(c). The error rates for the computer are listed in Table 5 in Appendix B3. In our $n = 3$ qubit experiments, we use the qubits with the lowest two-qubit error rates, namely the qubits labeled $(0, 1, 2)$. For $n = 5$ qubit experiments we use all qubits on the device.

2) RIGETTI ASPEN-M2

The Rigetti Aspen-M2 computer consists of 80 superconducting qubits arranged in a hexagonal lattice shown in Fig. 1(c). The error rates for the computer are listed in Table 7 in Appendix B3. We perform $n = 3$ qubit experiments on Rigetti Aspen-M2 using a line of qubits with relatively low two-qubit error rates, namely, the qubits labeled $(10, 17, 113)$ highlighted in Fig. 1(c). Due to limited device availability, we were only able to perform $n = 3$ qubit experiments on this computer.

3) IONQ HARMONY

The IonQ Harmony computer consists of 11 trapped ion qubits with all-to-all connectivity, shown in Fig. 1(c). Unlike the IBMQ Lima and Rigetti Aspen-M2 computers, at the time of performing experiments on IonQ Harmony, it was not possible to select which qubits to use when submitting jobs or to check which qubits were used after jobs were completed. The average one-qubit and two-qubit gate errors are, respectively, $\epsilon_{1Q} = 0.0029$ and $\epsilon_{2Q} = 0.0073$.

It is also worthwhile to note that, at the time of performing experiments, it was not possible (from the AWS platform) to disable compilation on IonQ Harmony, unlike on IBMQ Lima and on Rigetti Aspen-M2. Disabling compilation is important in error mitigation because techniques often insert gates that are logically trivial (e.g., $GG^\dagger$ in zero-noise extrapolation) or perform other modifications to produce

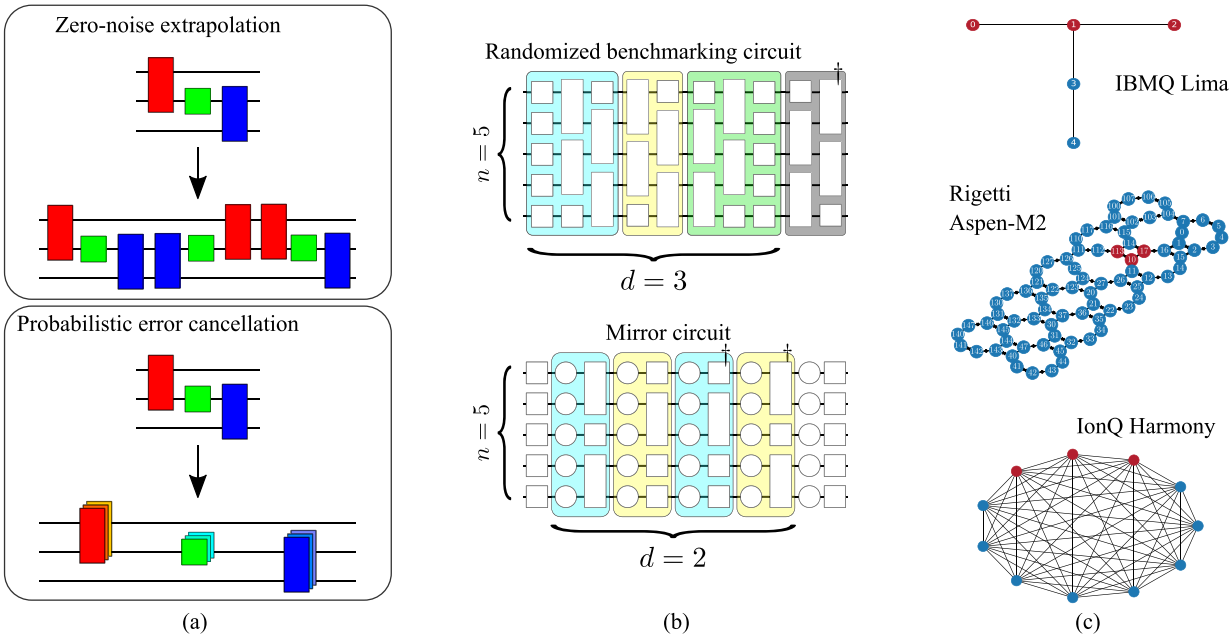

FIGURE 1. Overview of our method to assess the performance of quantum error mitigation in practice. An experiment consists of (a) a QEM technique, (b) a benchmark problem, and (c) a quantum computer. The result of an experiment is the improvement factor (Section II-A) μ_{QEM} defined in (6), potentially at various numbers of qubits n and/or circuit depths d. (a) Cartoon graphics of the two quantum error mitigation techniques we consider. In zero-noise extrapolation (Section II-B1), a circuit C is mapped to a set of noise-scaled circuits by adding gates that compile (without noise) to the identity. The expectation value is computed for each noise-scaled circuit, and the results are extrapolated to zero-noise with either linear or Richardson extrapolation. In probabilistic error cancellation (Section II-B2), each ideal (unitary) gate of a circuit is expressed in the noisy basis of the computer. A number of new circuits are sampled from this expansion and executed results are combined to produce the error-mitigated result. (b) Two benchmark problems we use in our experiments (Section II-C). An n-qubit, depth d mirror circuit C is defined by a single layer of Clifford gates (white squares), d Clifford layers, followed by their inverses (rectangles in colored boxes, daggers denote inverses) with intermediate random Pauli gates (circles in colored boxes), and a final layer of Pauli (white circles) and Clifford gates. This sequence produces a single bitstring $|z_C\rangle$, and we take $\hat{A} = |z_C\rangle\langle z_C|$ as the observable. An n-qubit, depth d randomized benchmarking circuit is defined by a random sequence of d elements of the n-qubit Clifford group and a final inverse such that the final state is $|0\rangle$, and we take $\hat{A} = |0\rangle\langle 0|$ as the observable. (c) Qubit coupling maps for the four quantum computers we perform experiments on (see Section II-D for device characteristics and error rates). Red nodes show qubits used for $n = 3$ qubit experiments. Qubit selection was not available on IonQ Harmony and so nodes are unlabeled. We also perform experiments on noisy quantum computer simulators (Section II-D5).

circuits that are meant to be run exactly as specified. To avoid these problems when running ZNE on IonQ Harmony, we add barriers of single-qubit infinitesimal rotations as described in Appendix B6. Due to limited device availability, we were only able to perform $n = 3$ qubit experiments on this computer.

4) IBMQ KOLKATA

To assess the performance of error mitigation on larger benchmark problem sizes (namely, $n = 12$ qubit experiments), we use the IBMQ Kolkata device based on the 27-qubit superconducting chip—see Appendix B3 for the coupling map (see Fig. 9) and error rates (see Table 6).

5) NOISY QUANTUM COMPUTER SIMULATORS

In addition to quantum computers, we also perform experiments on noisy quantum computer simulators (hereafter "noisy simulators"). There are two primary reasons for this. First, this allows us to compare how error mitigation performs with simple noise models relative to actual quantum computers. Second, using noisy simulators allows us to circumvent practical limitations like device availability to perform additional experiments.

We consider two classes of noisy simulators: 1) one which implements a simple noise model of single-qubit depolarizing noise after each gate and 2) one which is based on the error rates of a particular computer and so is meant to closely emulate that computer. The simple noise model we use is 1% depolarizing noise after each two-qubit gate, i.e., each two-qubit gate (CNOT) is replaced by (11) with $p = 0.01$. In addition to this simple noise model, we also use a noise model based on the error rates of the IBMQ Lima computer (referred to as FakeLima). This noisy simulator has the same topology as IBMQ Lima shown in Fig. 1(c) and includes inhomogeneous single-qubit gate errors, two-qubit gate errors, and measurement errors based on the device characteristics in Table 5. Similarly, we used IBMQ FakeKolkataV2 to classically simulate the hardware experiments performed on the real IBMQ Kolkata device. Its coupling map and error rates are reported in Fig. 9 and Table 6, respectively.

III. RESULTS

As described in Section II, we applied ZNE(L), ZNE(R), and PEC to the RB circuit and mirror circuit benchmarks on IBM, IonQ, and Rigetti quantum computers, as well as noisy quantum computer simulators. For each of these

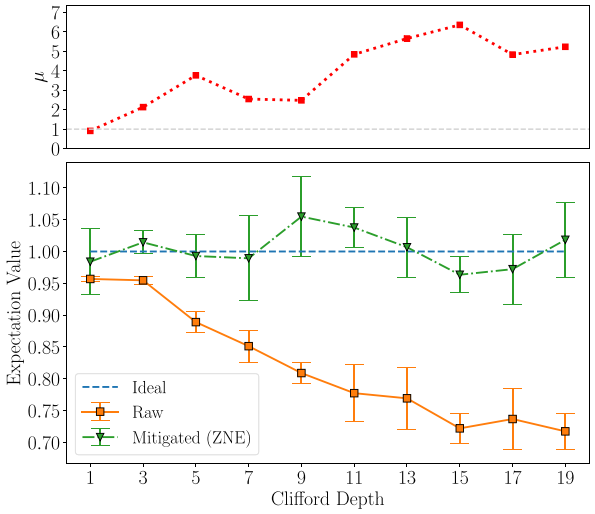

FIGURE 2. (Bottom panel) Unmitigated expectation values (orange squares) and the corresponding mitigated expectation values using ZNE(R) (green triangles) for $n = 3$ qubit RB circuits executed on IBMQ Lima. The size of the error bars for the "Raw" line is dictated by the standard deviation of the unmitigated values while the error bar sizes for the "Mitigated (ZNE)" line is the standard deviation of the mitigated values. For all depths, the ideal expectation value is equal to 1 (dotted line). (Top panel) The improvement factor (6) at each depth for the results in the bottom panel.

experiments, we compute the improvement factor (6) to quantify the performance of each error mitigation technique.

An example of the results from one particular experiment is shown in Fig. 2. Here, we show the result from applying ZNE(R) to $n = 3$ qubit RB circuits of various depths on IBMQ Lima. At each depth, we generate $|\mathcal{C}| = 4$ RB circuits and evaluate the expectation value with and without error mitigation using $t = 1$ trial, and use this to compute the improvement factor (6). As shown in Fig. 2, the "raw" (unmitigated) results diverge from the ideal (noiseless) expectation value as the depth d increases, while the ZNE(R) results are closer to the ideal expectation value but generally have higher variance. This is quantified in the improvement factor which here ranges from $\mu_{\text{ZNE(R)}} \simeq 1$ to $\mu_{\text{ZNE(R)}} \simeq 6$. In particular, all depths $d > 1$ show an improvement factor $\mu_{\text{ZNE(R)}} > 1$, indicating ZNE(R) was always beneficial to use in this example.

We show the results of all $n = 3$ experiments in Fig. 3. Here, results are arranged in a grid displaying error mitigation techniques and benchmark problems, and different colored markers in each subplot show results on different quantum computers, including noisy simulators. As a baseline for comparison, we consider a very simple noise model of 1% depolarizing noise (see Section II-D5), and we find as expected that this simple noise model generally produces the largest improvement factors in experiments. (The interesting exception is ZNE(R) for which IBMQ Lima shows the largest improvement factors.) On real quantum computers, there are additional sources of error including state preparation and measurement (SPAM) error as well as more complicated (in)coherent gate and crosstalk errors, so it is expected—as we see in the results—that these improvement factors are lower. However the improvement factors on the IBMQ Lima computer—as well as the improvement factors on the IBMQ Lima simulator which follow the real computer fairly closely—are still above $\mu = 1$, generally ranging between $\mu \simeq 1$ and $\mu \simeq 4$, indicating that error mitigation is beneficial on this device.

The improvement factors for PEC are lower, between $\mu \simeq 1$ and $\mu \simeq 2$, so PEC was generally less beneficial to run than ZNE, but still more beneficial than no error mitigation for most back ends (IBM, IonQ, and all simulators.). Moreover, we should take into account that PEC was applied assuming a very simplified noise model (depolarizing) and so we expect better performances when PEC is based on a more faithful noise characterization. On IonQ Harmony, there are several cases where $\mu < 1$, especially in ZNE(R) experiments, so ZNE(R) was generally worse to use than no error mitigation on this computer, while ZNE(L) was more beneficial than no error mitigation. Interestingly, most improvement factors on Rigetti Aspen-M2 are close to $\mu = 1$, so error-mitigated results were generally the same as unmitigated results on this computer. Recall that our improvement factor (6) normalizes by additional shots.

We repeat the same type of experiments using $n = 5$ qubits and show these results in a similar format in Fig. 4. Here we were unable to perform experiments on Rigetti Aspen-M2 or IonQ Harmony due to limited device availability. We see again in these results that the improvement factors for the simple 1% depolarizing noise model are generally the largest, as expected. The improvement factors on IBMQ Lima are comparable in magnitude to the $n = 3$ experiments, and in all cases, $\mu \geq 1$ so error mitigation was always beneficial. We also see that the IBMQ Lima simulator results follow the results of the real computer fairly closely, as in the $n = 3$ qubit experiment.

To further test the performance of QEM as the problem size increases, we perform $n = 12$ qubit experiments on both a hardware device and a noisy quantum computer simulator and show these results in Fig. 5. The hardware device is the 27-qubit IBMQ Kolkata computer and the noisy simulator is based on this Kolkata device (see Fig. 9 for the coupling map and Table 6 for the error rates). Here we see that improvement factors range from $\mu \simeq 1$ to $\mu \simeq 3$ indicating that error mitigation is still effective on larger problem sizes. The improvement factors for (parallel) randomized benchmarking circuits are higher than for mirror circuits in all cases, likely due to the fact that mirror circuits contain two-qubit gates across all edges. This is somewhat visible in $n = 3, 5$ qubit experiments but more accentuated here due to the larger problem size, and suggests that additional error mitigation on top of ZNE may be necessary to mitigate errors and crosstalk effects in larger applications. However, ZNE still performed better than the unmitigated experiments in all cases.

In the next subsections, we consider the dependence of the improvement factor defined in (6) with respect to different specific parameters: the number of shots (Section III-A), the

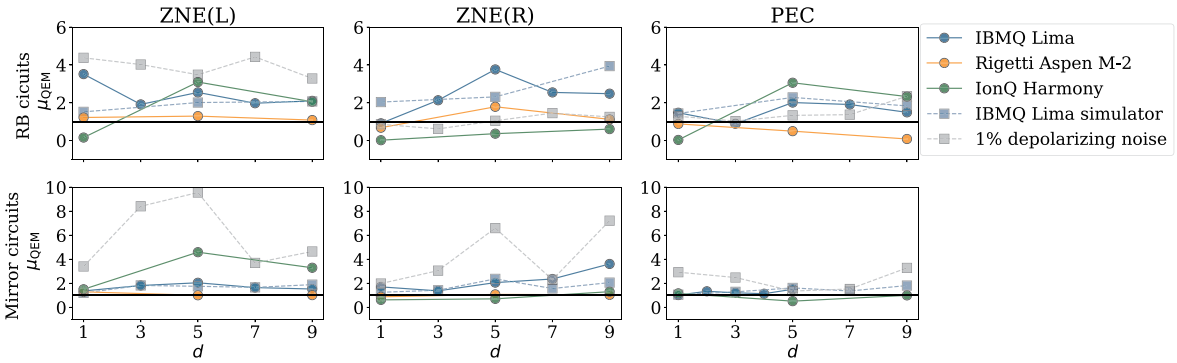

FIGURE 3. Improvement factor (6) results for $n = 3$ qubit experiments. From left to right, the quantum error mitigation techniques are ZNE with linear extrapolation, ZNE with Richardson extrapolation, and probabilistic error cancellation. The top panel shows improvement factors for over $|\mathcal{C}| = 4$ randomized benchmarking circuits, and the bottom panel shows results for $|\mathcal{C}| = 4$ mirror circuits. Circle markers show quantum computer results and square markers show noisy quantum computer simulator results. In most cases, improvement factors are highest for the 1% depolarizing noise model (gray squares), which is expected as this is the simplest noise model. Improvement factors on IBMQ Lima (blue circles) are almost always above $\mu = 1$. Improvement factors on Rigetti Aspen-M2 (orange circles) and IonQ Harmony (green circles) are frequently below $\mu = 1$ — notably for ZNE(R) RB circuits — indicating that error mitigation did not help in these experiments. Improvement factors from PEC are notably smaller than the ZNE experiments but are mostly above $\mu = 1$.

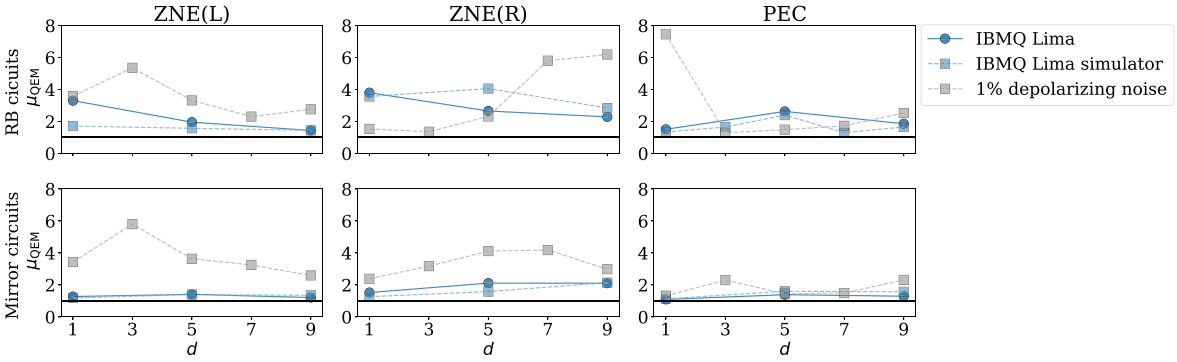

FIGURE 4. Improvement factor (6) results for $n = 5$ qubit experiments, in the same format as Fig. 3. In this case, we still see that improvement factors are usually highest on the 1% depolarizing noise simulator, as expected. The average improvement factors for ZNE are comparable but slightly lower than those of the $n = 3$ qubit experiments, whereas the average improvement factors for PEC are noticeably larger than those of the $n = 3$ qubit experiments.

observable weight (Section III-B), and the number of trials (Section III-C).

A. DEPENDENCE OF IMPROVEMENT FACTOR ON THE NORMALIZATION FACTOR c

In the hardware and numerical experiments presented in Figs. 2–5, we intentionally used precisely the same resources for estimating unmitigated and mitigated results. That is, we always computed improvement factors in which the cost normalization factor c, which appears in (6), is equal to 1. In Fig. 6 instead, we compute the unnormalized (top panel) and normalized (bottom panel) improvement factor evaluated at a different shot ratio N_{QEM}/N, for a simulated IBM device (FakeLima). It appears that, without normalization ($c = 1$), increasing the number of shots improves the results for the ZNE and PEC error mitigation techniques. However, when normalizing by the measurement cost, quantified as $c = \sqrt{N_{\mathrm{QEM}}/N}$, the improvement factor decreases.

From a theoretical point of view, there is not an obvious and unique choice for the normalization factor c, because the choice mainly depends on practical constraints. For example, in this work, we suggested using $c = \sqrt{N_{\mathrm{QEM}}/N}$, but different normalization factors have been proposed in the literature [43]. Moreover, one can imagine the extreme situation in which one just wants to benchmark the estimation accuracy for error-mitigated expectation values independently from the associated resource cost, such that the unnormalized improvement factor [(6) with $c = 1$)] can represent an appropriate metric.

B. DEPENDENCE OF IMPROVEMENT FACTOR ON THE LOCALITY OF THE OBSERVABLE

In this subsection, we study the dependence of the improvement factor on the number of qubits on which the estimated observable acts on, often referred to as the *weight* of the observable.

We consider a randomized benchmarking circuit $\mathcal{C}$ with $n = 12$ qubits and ideal measurement result $z_{\mathcal{C}} = 0 \ldots 0$, executed on a simulated IBM device (FakeKolkata). In all the experiments that we have previously presented (see, e.g., Fig. 5), we always considered the maximum-weight observable $A = |0 \ldots 0\rangle\langle 0 \ldots 0|$. Here instead, we define the

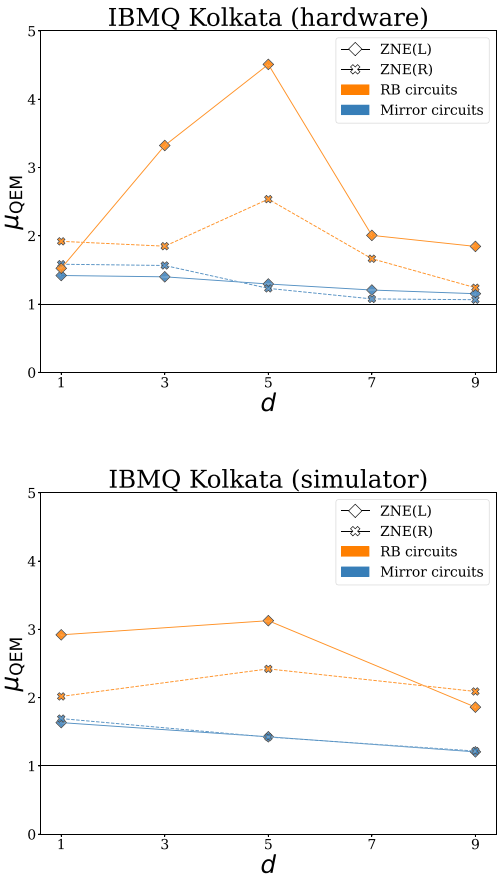

FIGURE 5. Improvement factor (6) results for $n = 12$ qubit experiments on both a quantum device and a noisy simulator based on the IBMQ Kolkata computer (see Fig. 9 for the coupling map and Table 6 for error rates).

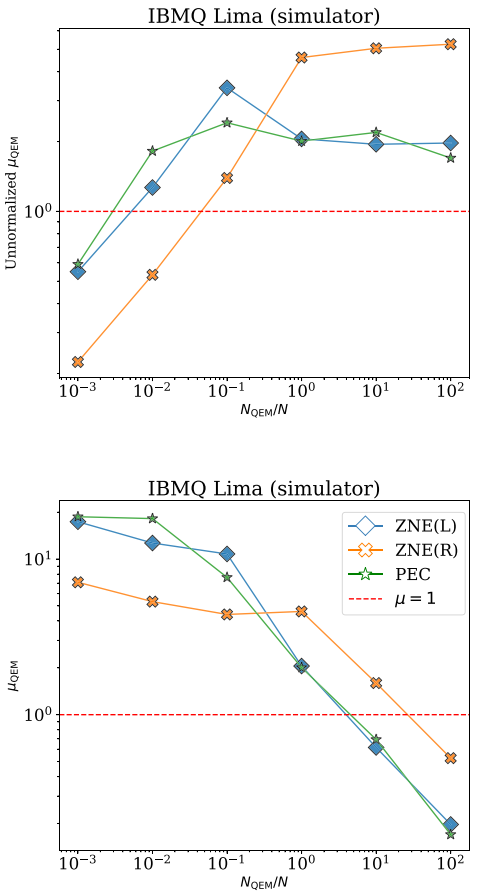

FIGURE 6. Improvement factor defined as in (6) and unnormalized, i.e., with $c = 1$ (top panel). Improvement factor normalized by the measurement cost quantified as $c = \sqrt{N_{\mathrm{QEM}}/N}$ where N is the maximum number of shots (bottom panel). In both panels, in the x-axis we report the shot ratio N_{QEM}/N, where N_{QEM} is the number of shots employed for error mitigation and N is the number of shots used without error mitigation. The benchmark is based on a randomized benchmarking circuit of depth $d = 5$ acting on $n = 3$ qubits and executed on the IBM FakeLima device.

following observables:

$$A_w = |0\ldots0\rangle\langle 0\ldots0|_{1:w} \otimes \mathbf{1}_{w+1:n}, \quad w = 1, 2, \ldots n \tag{13}$$

where $|0\ldots0\rangle\langle 0\ldots0|_{1:w}$ is the projector on the zero state of the first w qubits and $\mathbf{1}_{w+1:n}$ is the identity on the remaining qubits. By construction, the weight of A_w is w and, for a randomized benchmarking circuit, the ideal expectation value is 1 for all values of w.

Estimating a low-weight observable is typically easier with or without error mitigation. So we expect both the unmitigated and the mitigated estimation errors to increase with respect to w. However, since the improvement factor defined in (6) is given by the error ratio, it is hard to guess how μ_{QEM} depends on the observable weight.

Indeed, in Fig. 7 (top panel), we plot the improvement factor against w and we note a nontrivial behavior for different error mitigation techniques. On the other hand, in Fig. 7 (bottom panel), we observe that, as expected, the bare rmse associated with each error mitigation technique has a monotonically increasing dependence on w. Compared to ZNE, PEC seems to be more scalable with respect to the observable weight.

C. DEPENDENCE OF IMPROVEMENT FACTOR ON THE NUMBER OF TRIALS

In all the experiments executed on real hardware, we only considered a single trial $t = 1$, even if we averaged over multiple random circuits. In Fig. 8, instead, we show, through numerical simulations, how the improvement factor changes as we progressively average over more trials. We note that the improvement factors are more or less constant as the number of trials increases, with a small exception of Richardson extrapolation which appears to fluctuate for earlier numbers of trials but does become relatively constant after ten trials.

A possible explanation for the weak dependence of μ_{QEM} on the number of trials is that, even when using a single trial, the improvement factor formula in (6) is still an average over different instances of random circuits. Such an average over different circuits is probably enough to smooth the statistical fluctuations of the improvement factor.

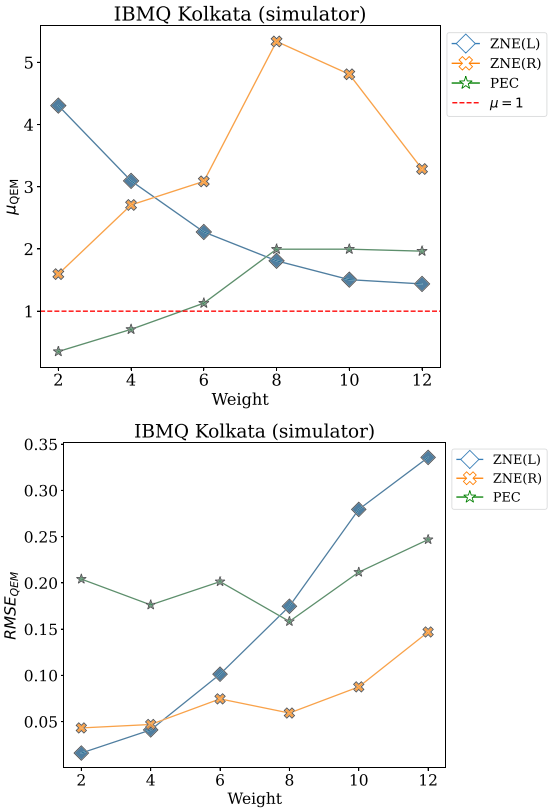

FIGURE 7. (Top panel) Improvement factor (6) associated to the observable A_w introduced in (13) as a function of w, i.e., the number of qubits on which A_w acts nontrivially. (Bottom panel) same results but reporting the mitigated root-mean-squared error (without dividing by the unmitigated error). The numerical simulations are performed on the IBM FakeKolkata device, for a randomized benchmarking circuit of depth $d = 5$ acting on $n = 12$ qubits.

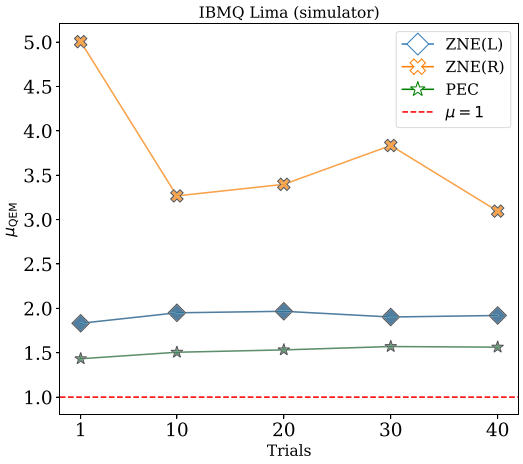

FIGURE 8. Improvement factor (6) as a function of the number of trials. The benchmark circuit used in this figure is a randomized benchmarking circuit of depth $d = 5$ acting on $n = 3$ qubits and executed on the IBM FakeLima device.

IV. DISCUSSION

A. POSITIVE FEATURES OF OUR WORK

A positive aspect of our experiments is that we apply error mitigation techniques "out-of-the-box," i.e., we do not tailor any techniques to the benchmark problems or computers we consider. Indeed, all experiments were performed with high-level API calls to QEM software [27] (see Appendix B for more on the implementation details). While tailoring techniques to specific experiments are likely to provide better results and are advisable in most applications, our approach gives a picture of what can be expected from QEM in general applications.

Another positive feature of our work is the comparison of results across several computers. This experimentally verifies that error mitigation can indeed be viewed as an algorithmic or software method independent of hardware, but our results also emphasize that the performance of error mitigation depends on the underlying computer. A clear example that illustrates this is zero-noise extrapolation with very deep circuits such that the final state is approximately the maximally mixed state. In this scenario, scaling noise further does not produce any signal from which one can extrapolate, so ZNE has no advantage relative to no error mitigation. On a more accurate quantum computer, however, the final state may not be maximally mixed and noise may be able to be scaled with small-scale factors. On the opposite limit, if a quantum computer is already very accurate, the improvement factor due to error mitigation is necessarily small. In fact, in the limit of very weak noise, the bias of expectation values is negligible compared to the statistical variance which is typically not reduced by error mitigation (actually, it is often increased by it).

Beyond experimental results, our work introduces a quantitative, problem-independent, and resource-normalized measure of the improvement of QEM, the improvement factor, namely, (6). This is a natural and empirically motivated measure that introduces a standardized metric for measuring and comparing error-mitigated quantum computer performance. For this reason, we expect the improvement factor metric to be used in other future experiments, beyond this work. We have shown the relation of our metric to the metric in [9] so that results can be compared, and we encourage the use of quantitative metrics in future error mitigation work to continue this effort. Finally, we incorporated the notion of normalizing the additional resources (namely, the cost factor c) in our definition of the improvement factor, and experimentally showed error mitigation can still be beneficial even when adjusting by these extra resource requirements.

B. LIMITATIONS OF OUR WORK

While the cost factor defined in (5) normalizes by the additional shots used in QEM, it does not normalize by additional qubits, gates, or circuits. While the error mitigation techniques we used in this work do not increase the number of qubits in executed circuits, other techniques do require more qubits in executed circuits, and accounting for this resource is important to understand the value of QEM.

IEEE Transactions on Quantum Engineering

In more detail, the techniques we use can increase both the number of gates and the number of circuits executed in an experiment. In particular, in ZNE, the number of gates in each circuit is primarily increased but the number of circuits is only slightly increased, while in PEC the number of gates in each circuit is essentially the same as the original circuit but the number of circuits to execute is significantly higher. The optimal way to account for these additional resources is not immediately obvious, for example, whether to count additional circuits separately or to only count the total number of gates in all circuits. Similarly, even if additional circuits use the same number of qubits as the original circuit, they could be executed simultaneously by using additional qubits, so there is some subtlety concerning the most appropriate way to normalize for these space-time tradeoffs. Ultimately, these questions are likely to depend on the particular computer being used, but the variety and flexibility of available error-mitigating techniques should be encouraging to builders and users of the many different quantum computer architectures being proposed.

Another potential resource in QEM is preprocessing/noise characterization. For example, in PEC, one needs to determine the noise basis of the computer, and in measurement error mitigation one needs to characterize the confusion matrix. However, these examples and others usually do not grow with the size of the problem. Thus, they are likely less important to account for in the improvement factor. (Recall that we assumed a particular noise model for PEC and did not perform gate characterization, so preprocessing cost is not present in our improvement factor results.) Applying "out-of-the-box" error mitigation techniques at the gate level may be paired with tailored and more complex noise models going beyond error calibration information produced by hardware providers [5], [45].

Although we experimentally evaluate QEM more generally than current literature (see Table 1), we still considered just two error mitigation techniques out of (roughly) dozens proposed in the literature. Additionally, both benchmark problems we used are based on random circuits—while we expect our results to extend to structured circuits (say for time evolution), this needs to be experimentally verified. Due to limited device availability (i.e., which computers and how much computer time we had access to), we were only able to perform experiments on up to $n = 12$ qubits. This is fairly typical for error mitigation experiments (see Table 1), and we used noisy simulators to test the performance of error mitigation on larger problem sizes, but experiments on larger computers are still desirable. Last, we only considered experiments with a single error mitigation technique. Experiments composing two or more error mitigation techniques, for example, zero-noise extrapolation with dynamical decoupling and measurement error mitigation, will likely yield the largest improvement in applications. We leave quantifying this improvement (again normalized by additional resources used) and other points mentioned here for future work.

C. RELATIONSHIP TO THE LITERATURE

Our work adds a significant number of QEM experiments relative the current literature (see Table 1). Notably, our work introduces a quantitative, problem-independent, and resource-normalized measure of the improvement of QEM, and we compute this quantity on multiple quantum computers. With the notable exception [9], our work goes significantly beyond most experimental studies of QEM which typically consider one technique for a specific experiment and do not explicitly evaluate a quantitative improvement of QEM.

D. FUTURE OUTLOOK OF QEM

Based on our results and other experiments in the literature using error mitigation, we expect QEM to be an essential component of virtually all experiments on "NISQ" computers [34], where we can take "NISQ" to roughly mean computers with up to $n \sim 10^2$ qubits capable of implementing $d \sim 10^3$ two-qubit gates. Indeed, depending on how much overhead is required for error correction, error mitigation techniques may continue to be important at even larger scales.

Abstractly, QEM can be viewed as combining NISQ computers with classical processors, and we have shown that this combination is still beneficial even when normalized by additional resources used. This combination is likely to be most beneficial when applied to problems that are classically hard [1]. In this setting, a quantum computer is used to get a rough solution to a hard problem, and a classical computer is used to improve the accuracy of the solution. We expect that combining the strengths of both devices in this manner will be necessary for solving problems too hard for either device to solve individually.

Furthermore, it is likely that QEM will play an important role beyond NISQ computations, namely in error-corrected computations. Although QEM and QEC are sometimes thought of as separate techniques due to their different resource requirements and generality, they are similar in that both are algorithmic or software techniques to deal with errors in quantum computers. For example, dynamical decoupling—largely considered to be a QEM technique in many settings—has been an important technique in recent quantum error correction experiments [3] to improve the fidelity of data qubits while syndrome measurements are performed. Further, [41] provides a more theoretical discussion about the application of QEM in fault-tolerant quantum computing. We anticipate additional work connecting error mitigation to error correction at both the theoretical and experimental levels. Our results and the experimental framework for obtaining these results [27] provide some foundation for progress in this direction.

V. CONCLUSION

In this work, we experimentally tested the performance of QEM. Using an empirically motivated metric that normalizes by the number of additional resources used in a QEM

technique, we quantified the improvement of error mitigation using a variety of benchmark problems and quantum computers. In particular, we tested zero-noise extrapolation and probabilistic error mitigation on two benchmark problems and three quantum computers. The largest of such error mitigation benchmarks involved quantum circuits acting on 12 qubits with more than 100 two-qubit gates and more than 600 single-qubit gates. Our results show that error mitigation is on average more useful than no error mitigation, even when normalizing by the additional resources used and applying "out-of-the-box" error mitigation, i.e., not tailoring the technique to the specific benchmark problem or the specific quantum computer. While these latter points are likely to provide further improvements and are encouraged in applications, our results provide a general picture of what can be expected from QEM in practice.

Our definition of the improvement factor is, together with the very recent proposal of [43], one of the first quantitative metrics to normalize by additional resources used in error mitigation, and we encourage the adoption of this or similar metrics in future work. It is also of interest to expand this metric for resources we do not account for here—for example, additional qubits and gates—to more fully understand and quantify the value of QEM in real experiments. This also can be considered with error mitigation techniques we did not use here—for example, dynamical decoupling) [14], [33], [53], [54], [58], Clifford data regression [6], [26], and noise-extended probabilistic error cancellation [32]. Additional experimental and theoretical results of this nature will help to further the progress made in this work and better understand the value of error mitigation in the larger context of quantum computing and quantum error correction.

Note added: While preparing our manuscript, we noticed a recent review of QEM [7] which is similar in scope but discusses QEM from a more theoretical rather than experimental perspective as in this article.

APPENDIX A
NOTATION

A summary of notation is shown in Table 4.

APPENDIX B
IMPLEMENTATION DETAILS

In this appendix, we discuss the implementation of the experiment and provide more details pertaining to how these experiments were programmatically carried out.

A. EXPERIMENT SETUP

For a given experimental run, the user specifies whether to run on a quantum hardware or simulator device, the platform to target (IBM, IonQ, or Rigetti) covered in B3, the error mitigation method to apply (PEC or ZNE) covered in B4, and the circuit type to consider (RB or mirror) covered in B5.

Once we obtain either our RB or mirror circuit C, we define an executor function that takes the input circuit and returns an

TABLE 4 Summary of Notation

C	Quantum circuit (unitary)
ρ	Final state of circuit C, $\rho = C\|0\rangle\langle 0\|C^\dagger$
$\hat{A}$	Hermitian observable
n	Number of qubits
d	Depth of circuit
N	Number of shots (samples)
A	Ideal expectation value $\mathrm{tr}[\rho\hat{A}]$
$\mathcal{E}$	Channel of a noisy computer (simulator)
A'	Noisy expectation value $\mathrm{tr}[\mathcal{E}(\rho)\hat{A}]$
QEM	Quantum error mitigation technique
A_{QEM}	Error-mitigated expectation value
N_{QEM}	Total number of shots used in the QEM technique
$\mathcal{C}$	Set of (benchmark) circuits
$\hat{\mathcal{A}}$	Set of (benchmark) observables
μ_{QEM}	Improvement factor Eq. (6) of QEM technique
ZNE	Zero-noise extrapolation
ZNE(L)	ZNE with linear extrapolation
ZNE(R)	ZNE with Richardson extrapolation
PEC	Probabilistic error cancellation

TABLE 5 IBMQ Lima Error Rates for Each Qubit in the Coupling Map From Fig. 1(c); Here, ϵ_{1Q}, ϵ_{CX}, and ϵ_M Represent the Single-Qubit $\sqrt{X}$-Gate Error, the Average Two-Qubit CNOT Error, and the Readout Assignment Error, Respectively [21]

Qubit	$\epsilon_{1\mathrm{Q}}$	ϵ_{CX}	ϵ_M
0	9.028×10^{-4}	1.026×10^{-2}	2.740×10^{-2}
1	6.452×10^{-4}	1.083×10^{-2}	1.510×10^{-2}
2	6.016×10^{-4}	7.375×10^{-3}	1.700×10^{-2}
3	2.523×10^{-4}	1.570×10^{-2}	2.360×10^{-2}
4	6.167×10^{-4}	1.655×10^{-2}	4.410×10^{-2}

TABLE 6 IBMQ Kolkata Error Rates for Each Qubit Used in Our $n = 12$ Qubit Experiments (See Fig. 9)

Qubit	$\epsilon_{1\mathrm{Q}}$	ϵ_{CX}	ϵ_M
0	1.640×10^{-4}	3.997×10^{-3}	1.820×10^{-2}
1	1.339×10^{-4}	5.737×10^{-3}	1.850×10^{-2}
4	1.660×10^{-4}	6.636×10^{-3}	2.750×10^{-2}
7	2.027×10^{-4}	1.259×10^{-2}	2.270×10^{-2}
10	4.948×10^{-4}	1.397×10^{-2}	1.320×10^{-2}
12	2.380×10^{-4}	9.326×10^{-3}	8.500×10^{-3}
15	2.626×10^{-4}	4.086×10^{-2}	5.000×10^{-3}
18	2.248×10^{-4}	1.052×10^{-2}	6.200×10^{-3}
21	1.772×10^{-4}	6.663×10^{-3}	1.350×10^{-2}
23	2.221×10^{-4}	5.340×10^{-3}	8.100×10^{-3}
24	2.858×10^{-4}	5.027×10^{-1}	1.360×10^{-2}
25	5.048×10^{-4}	3.368×10^{-1}	6.600×10^{-3}

We use a noisy simulator based on these error rates to perform experiments. Here, $\epsilon_{1\mathrm{Q}}$, ϵ_{CX}, and ϵ_{M} represent the single-qubit $\sqrt{X}$-gate error, the average two-Qubit CNOT error, and the readout assignment error, respectively [21].

TABLE 7 Rigetti Aspen-M2 Measures of the Average Two-Qubit CNOT Error (ϵ_{CX}) for the Qubit Edge and Relative Readout Fidelity Rates for the Qubits in the Device That We Use in Our Experiments: Accessed From [4]

Qubit specs		Edge specs	
Qubit	Readout fidelity	Edge	ϵ_{CX}
10	99.3%	10-17	3.79×10^{-3}
17	98.2%	10-113	4.58×10^{-3}
113	94.7%		

IEEE Transactions on Quantum Engineering

TABLE 8 Summary of Experiments Performed

Platform	Computer	QEM	Extrapolation	Circuit	Qubits	Simulator
IBM	Lima	ZNE	Richardson	RB	3	No
IBM	Lima	ZNE	linear	RB	3	No
IBM	Lima	PEC	–	RB	3	No
IBM	Lima	ZNE	Richardson	RB	5	No
IBM	Lima	ZNE	linear	RB	5	No
IBM	Lima	PEC	–	RB	5	No
IBM	Lima	ZNE	Richardson	Mirror	3	No
IBM	Lima	ZNE	linear	Mirror	3	No
IBM	Lima	PEC	–	Mirror	3	No
IBM	Lima	ZNE	Richardson	Mirror	5	No
IBM	Lima	ZNE	linear	Mirror	5	No
IBM	Lima	PEC	–	Mirror	5	No
IBM	Kolkata	ZNE	Richardson	RB	3	No
IBM	Kolkata	ZNE	linear	RB	3	No
IBM	Kolkata	ZNE	Richardson	Mirror	3	No
IBM	Kolkata	ZNE	linear	Mirror	3	No
IonQ	Harmony	ZNE	Richardson	RB	3	No
IonQ	Harmony	ZNE	linear	RB	3	No
IonQ	Harmony	PEC	–	RB	3	No
IonQ	Harmony	ZNE	Richardson	Mirror	3	No
IonQ	Harmony	ZNE	linear	Mirror	3	No
IonQ	Harmony	PEC	–	Mirror	3	No
Rigetti	Aspen-M2	ZNE	Richardson	RB	3	No
Rigetti	Aspen-M2	ZNE	linear	RB	3	No
Rigetti	Aspen-M2	PEC	–	RB	3	No
Rigetti	Aspen-M2	ZNE	Richardson	Mirror	3	No
Rigetti	Aspen-M2	ZNE	linear	Mirror	3	No
IBM	FakeLima	ZNE	Richardson	RB	3	Yes
IBM	FakeLima	ZNE	linear	RB	3	Yes
IBM	FakeLima	PEC	–	RB	3	Yes
IBM	FakeLima	ZNE	Richardson	RB	5	Yes
IBM	FakeLima	ZNE	linear	RB	5	Yes
IBM	FakeLima	PEC	–	RB	5	Yes
IBM	FakeLima	ZNE	Richardson	Mirror	3	Yes
IBM	FakeLima	ZNE	linear	Mirror	3	Yes
IBM	FakeLima	PEC	–	Mirror	3	Yes
IBM	FakeLima	ZNE	Richardson	Mirror	5	Yes
IBM	FakeLima	ZNE	linear	Mirror	5	Yes
IBM	FakeLima	PEC	–	Mirror	5	Yes
IBM	FakeKolkataV2	ZNE	Richardson	RB	12	Yes
IBM	FakeKolkataV2	ZNE	linear	RB	12	Yes
IBM	FakeKolkataV2	ZNE	Richardson	Mirror	12	Yes
IBM	FakeKolkataV2	ZNE	linear	Mirror	12	Yes
AWS	1% depolarizing noise	ZNE	Richardson	RB	3	Yes
AWS	1% depolarizing noise	ZNE	linear	RB	3	Yes
AWS	1% depolarizing noise	PEC	–	RB	3	Yes
AWS	1% depolarizing noise	ZNE	Richardson	RB	5	Yes
AWS	1% depolarizing noise	ZNE	linear	RB	5	Yes
AWS	1% depolarizing noise	PEC	–	RB	5	Yes
AWS	1% depolarizing noise	ZNE	Richardson	Mirror	3	Yes
AWS	1% depolarizing noise	ZNE	linear	Mirror	3	Yes
AWS	1% depolarizing noise	PEC	–	Mirror	3	Yes

expectation value $\langle A \rangle$ where $A = |z\rangle\langle z|$ for each circuit and where z denotes the correct bitstring.

The circuits may contain gates that are not in the supported gatesets for the hardware device we are targeting to run on. In this case, we compile the gates in the circuit to gates in the supported gateset (more on this process in B5.)

We then loop over each Clifford depth and within this loop, iterate over the number of trials we want to perform at each depth. For each iteration within our trial, we calculate the result of applying our error mitigation method. The resulting data is saved. Further information on the saved data can be found in B2.

B. SOFTWARE AND EXPERIMENT DATA

Our experiments were carried out using Python 3.9. The error mitigation methods of PEC and ZNE were applied via

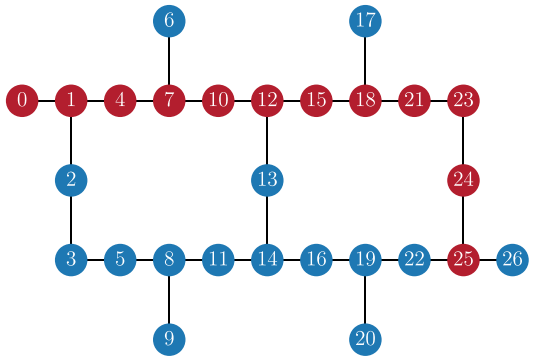

FIGURE 9. Coupling map for the 27-qubit IBMQ Kolkata computer. We use this device (see Table 6 for its properties) to perform $n = 12$ qubit error mitigation experiments on the red qubits. Results are shown in Fig. 5.

version `0.18.0` of the Mitiq Python software package. The libraries Cirq (version `1.0.0`), `amazon-braket-sdk` (version `1.25.2`), and Qiskit (version `0.38.0`) were used to specify the circuits for our experiments. Further information on the Mitiq package can be found on the official GitHub repository[2] as well as in [27].

The data obtained from our experiments and used to generate the plots in this work can be found in the `data` directory

```
data/TYPE/QEM/CIRCUIT/PLATFORM/
```

where `TYPE` $\in$ {`hardware`, `software`} describes whether the experiment was run on either an actual quantum device or a simulator, `QEM` $\in$ {`pec`, `zne`} describes the error mitigation method that was applied, `CIRCUIT` $\in$ {`mirror`, `rb`} describes the circuit type considered, and where `PLATFORM` $\in$ {`ibmq`, `ionq`, `rigetti`, `depolarizing`} describes on which platform the experimental data was obtained from.

Contained in each such directory is a subfolder with the following form:

```
PLATFORM_QEM_CIRCUIT_QUBITS_MIN_
MAX_SHOTS_TRIALS
```

where `QUBITS` is the number of qubits used in the experiment, `MIN` is the minimum Clifford depth, `MAX` is the maximum Clifford depth, `SHOTS` is the total number of shots used in the experiment (this is 10 000 for all of our experiments) and `TRIALS` is the total number of trials carried out per experiment (this is 4 for all of our experiments).

In each such subfolder is a listing of files with the following prefixes.

1) `cnot_counts`: The number of CNOT gates in the circuit.
2) `noise_scaled_expectation_values`: Noise-scaled expectation values (for ZNE only).
3) `noisy_values`: The nonscaled noisy expectation values (prior to applying error mitigation).
4) `oneq_counts`: The number of circuit instructions (minus the number of CNOT operations).
5) `true_values`: The ideal values (these are always equal to 1).
6) `mitigated_values`: The error-mitigated values.

Each row represents the value obtained at the depth corresponding to the index and each column represents the data obtained for a given trial.

Running the software in [52] that is responsible for capturing quantum device hardware experiment data requires possessing an AWS Braket account (for IonQ and Rigetti) an IBM Quantum account (for IBM). Running the software on exclusively quantum simulators can be done without any such account access.

To run the software on a simulator device the variable `use_noisy_simulator` should be set to `True` (and alternatively, `False` if the desire is to run on quantum device hardware). Setting the `mitigation_type` variable to either `pec` and `zne` runs PEC or ZNE error mitigation, respectively. The type of circuit to use can be set via the variable `circuit_type` to either `rb` for randomized benchmarking circuits or `mirror` for mirror circuits. Specifying the target platform can be done by setting the `hardware_type` variable to either `ibmq`, `iong`, or `rigetti` for IBM, IonQ, or Rigetti, respectively.

C. QUANTUM COMPUTING DEVICE PLATFORMS

Access to the Rigetti Aspen-M2 and IonQ Harmony hardware devices were provided via the Amazon Braket service on AWS. Hardware information pertaining to qubit topology, error rates, etc., was obtained via the Amazon Braket API. Access to the IBM hardware Lima device and IBM FakeLima and FakeKolkataV2 simulator devices were provided by the IBM Quantum Compute Resources page [21].

D. ERROR MITIGATION

The PEC and ZNE error mitigation methods were applied using the Mitiq software package.

1) ZNE

The Mitiq package employs local folding [19] and global folding [45] as gate-level noise scaling methods for ZNE. In this work, we used global folding.

Global folding increases the effective length of the quantum circuit by compiling the input circuit with a larger number of gates. Each set of layers in the circuit is replaced by $GG^\dagger G$. Since $GG^\dagger = I$, this has no effect when running our circuit on an ideal simulator. However, in the case where one uses a noisy device, this increases the noise and effective gate errors of the computation. An arbitrary example that depicts the global folding technique is shown in Fig. 10.

[2]https://github.com/unitaryfund/mitiq

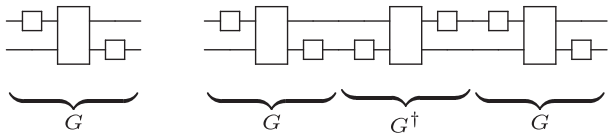

FIGURE 10. Example of the global folding technique applied to an arbitrary circuit. For a given collection of gates denoted by G, we increase the effective length of the overall circuit by creating $GG^{\dagger}G$.

```
import qiskit
from mitiq.zne.scaling import fold_global

# Define a quantum circuit.
qubits = qiskit.QuantumRegister(2)
circuit = qiskit.QuantumCircuit(qubits)
circuit.h(0)
circuit.cnot(0, 1)

# Apply global folding.
scaled_circuit = fold_global(circuit,
    scale_factor=3)
```

LISTING 1: Apply global folding to a circuit in Mitiq.

```
from mitiq import zne

zne_value = zne.execute_with_zne(
    circuit,
    execute,
    scale_noise=zne.scaling.fold_global,
    factory=zne.inference.RichardsonFactory
        (scale_factors=[1, 2, 3])
)
```

LISTING 2: Applying ZNE in Mitiq using Richardson extrapolation with noise scale factors $\lambda_i \in \{1, 2, 3\}$.

An example of how to make use of global folding in Mitiq is provided in Listing 1 that makes use of $\lambda = 3$ being a scale factor. Note that one can select $\lambda = 1$, which corresponds to note performing any folding, whereas selecting $\lambda = 3$ folds all of the gates in the circuit. For any scale factors $\lambda > 3$ in Mitiq, this folds some or all gates in the circuit.

For ZNE, we applied linear and Richardson extrapolation methods provided as factory objects in Mitiq as `LinearFactory` and `RichardsonFactory`, respectively (Listing 2).

2) PEC

To apply probabilistic error cancellation [15], [50], [57] we use the associated Mitiq module. We extract the two-qubit error probability p_{2Q} from the back end properties as reported by the hardware vendor. We use this information together with the utilities in `mitiq.representations` to generate the quasi-probability representations for all the two-qubit operations acting on neighboring qubits of the quantum processor. We store the result as a list (`representations`) of `mitiq.pec.OperationRepresentation` objects. After this preliminary step, we can obtain all the error-mitigated expectation values as described in Listing 3.

```
from mitiq import pec

pec_value = pec.execute_with_pec(
    circuit,
    execute,
    representations,
    num_samples=num_samples,
    random_state=local_seed,
)
```

LISTING 3: Apply PEC using Mitiq.

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

def get_circuit(depth, seed):
    circuit = rb.
        randomized_benchmarking_seq(
        length_vector=[depth],
        rb_pattern=rb_pattern,
        group_gates="0",
        rand_seed=seed,
    )[0][0][0]
```

LISTING 4: Defining RB circuits in Mitiq.

```
from mitiq.benchmarks import
    generate_mirror_circuit

def get_circuit(depth, seed):
    circuit, correct_bitstring =
        generate_mirror_circuit(
        nlayers=depth,
        two_qubit_gate_prob=1.0,
        connectivity_graph=computer,
        two_qubit_gate_name="CNOT",
        seed=seed,
        return_type=return_type,
    )
```

LISTING 5: Defining mirror circuits in Mitiq.

E. CIRCUITS

In order to construct our RB circuits, we define an RB pattern by splitting the qubits into 2-qubit pairs. We then generate a generic RB sequence via the Qiskit library. If the circuit is to be run on either Rigetti or IonQ, we perform a conversion of the Qiskit circuit to a Braket circuit (Listing 4).

We generate mirror circuits via the `generate_mirror_circuit` function from Mitiq. Mirror circuits parameterize the number of random Clifford layers to be generated (Listing 5).

1) CIRCUIT COMPILATION

At the time of this writing, both Rigetti and IonQ hardware support the option of *verbatim compilation*; a method that directs the compiler to run the specified circuit exactly as defined without adding any modifications. We attempt to disable automatic compilation by the platform service or QPU providers in order to have as much control as possible on the compiled circuit and hence error mitigation scaling.

The usage of verbatim compilation requires that every gate in the circuit is a gate that is natively supported by the hardware it is running on. For Aspen-M2, the native gate set is {RX, RZ, CPHASE, CZ, XY}. In our `compile_to_rigetti_gateset` function, we iterate through every instruction in our circuit and compile all of the gates into equivalent ones that are in the supported native gateset. As IonQ only recently added support for verbatim compilation, we were not able to take advantage of this option for the IonQ Harmony experiments.

2) TASK BATCHING

The AWS braket platform allows for task batching; the ability to launch jobs in parallel. For the majority of our experiments, serial execution within the allotted device time windows was sufficient to carry out a complete run of our experiment. One exception to this was the PEC experiments on Rigetti and IonQ hardware. In order to ensure these experiments ran within the allotted device time window, we needed to make use of the batching functionality.

F. ROTATION BARRIERS

Many hardware back ends internally optimize circuits before actually running physical gates on a quantum processor. This can be a problem for some error mitigation techniques. For example, in ZNE we want to run circuits whose depth is intentionally increased by unitary folding $G \to GG^\dagger G$ (see Fig. 10). However, if the internal optimizer of a back end detects that $G^\dagger G = I$, it will simplify the unitary folding structure such that any noise scaling effect is lost. This is a relevant practical issue for gate-level ZNE.

The best way to avoid this effect is to optimize circuits before applying ZNE and to switch off any further circuit optimizations on the back end side. When this is not possible, a simple work-around is the addition of barriers of infinitesimal gates that generate a negligible unitary effect on the quantum state but that can block the action of circuit optimizers.

In practice, in our experiments we apply ZNE by using a slightly modified version of the unitary folding rule

$$G \to GR_1\, G^\dagger R_2\, G \tag{B1}$$

where G is the circuit acting on n qubits and $R_j = [R_x(\epsilon_{x,j})R_y(\epsilon_{y,j})R_z(\epsilon_{z,j})]^{\otimes n}$ is the tensor-product of n infinitesimal rotations. For each circuit block of the unitary folding structure, we apply a rotation barrier. The way in which the small rotation angles are chosen is quite arbitrary as long as they are sufficiently small but nonzero. In our experiments, we randomly sample between two fixed small angles ($\pm 10^{-4}$) as reported in Listing 6.

Note, that all our ZNE experiments on IonQ hardware have been done via the AWS cloud platform before *verbatim compilation* became available. Today *verbatim compilation* of quantum circuits into native gates is supported for both IonQ and Rigetti devices. Therefore, the work-around of applying rotation barriers is probably not necessary anymore to reproduce similar experiments.

```python
import cirq
import numpy as np

def make_rotation_barrier(
    circuit,
    delta=0.0001,
):
    """Returns a barrier of infinitesimal
       rotations."""
    qubits = list(circuit.all_qubits())
    delta_x = np.random.choice([1.0, -1.0])
        * delta
    delta_y = np.random.choice([1.0, -1.0])
        * delta
    delta_z = np.random.choice([1.0, -1.0])
        * delta
    barrier = cirq.Circuit()
    for q in qubits:
        barrier.append(cirq.rx(delta_x)(q))
        barrier.append(cirq.ry(delta_y)(q))
        barrier.append(cirq.rz(delta_z)(q))
    return barrier
```

LISTING 6: Function returning a layer of infinitesimal rotations. Each layer is applied as a barrier between circuit blocks as shown in (B1).

ACKNOWLEDGMENT

The authors would like to thank E. Hickman for proposing, in a public discussion on the Mitiq Discord channel, the idea of using small rotations to block back end compilation. The authors would also like to thank D. Wang for running the IBM Kolkata hardware experiments. The authors would like to thank IBM, Rigetti, and IonQ for providing access to their quantum computers; the AWS Braket team for facilitating access to Rigetti and IonQ computers through the AWS Braket platform. The views expressed in this article are those of the authors and do not reflect those of AWS, IBM, IonQ, or Rigetti.

CODE AND DATA AVAILABILITY

Code and data are available upon reasonable HTTPS request to https://github.com/unitaryfund/research/.

REFERENCES

[1] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019, doi: 10.1038/s41586-019-1666-5.

[2] Google Quantum AI, "Exponential suppression of bit or phase errors with cyclic error correction," *Nature*, vol. 595, pp. 383–387, 2021, doi: 10.1038/s41586-021-03588-y.

[3] Google Quantum AI, "Suppressing quantum errors by scaling a surface code logical qubit," 2022, *arXiv:2207.06431*, doi: 10.48550/arXiv.2207.06431.

[4] Amazon Web Services, Amazon Braket, 2020. [Online]. Available: https://quantumcomputing.stackexchange.com/questions/25822/how-to-properly-cite-amazon-braket

[5] E. den van Berg, Z. K. Minev, A. Kandala, and K. Temme, "Probabilistic error cancellation with sparse Pauli-Lindblad models on noisy quantum processors," 2022, *arXiv:2201.09866*, doi: 10.48550/arXiv.2201.09866.

[6] P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, "Error mitigation with Clifford quantum-circuit data," *Quantum*, vol. 5, p. 592, 2021, doi: 10.22331/q-2021-11-26-592.

[7] Z. Cai et al., "Quantum error mitigation," 2022, *arXiv:2210.00921*, doi: 10.48550/arXiv.2210.00921.

[8] A. 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. 3, 2019, Art. no. 032328, doi: 10.1103/PhysRevA.100.032328.

[9] C. Cirstoiu, S. Dilkes, D. Mills, S. Sivarajah, and R. Duncan, "Volumetric benchmarking of error mitigation with qermit," 2022, *arXiv:2204.09725*, doi: 10.22331/q-2023-07-13-1059.

[10] A. D. Córcoles et al., "Process verification of two-qubit quantum gates by randomized benchmarking," *Phys. Rev. A*, vol. 87, no. 3, 2013, Art. no. 030301, doi: 10.1103/PhysRevA.87.030301.

[11] P. Czarnik, M. McKerns, A. T. Sornborger, and L. Cincio, "Improving the efficiency of learning-based error mitigation," 2022, *arXiv:2204.07109*, doi: 10.48550/arXiv.2204.07109.

[12] J. I. Colless et al., "Computation of molecular spectra on a quantum processor with an error-resilient algorithm," *Phys. Rev. X*, vol. 8, no. 1, 2018, Art. no. 011021, doi: 10.1103/PhysRevX.8.011021.

[13] A R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A*, vol. 54, no. 2, 1996, Art. no. 1098, doi: 10.1103/PhysRevA.54.1098.

[14] P. Das, S. Tannu, S. Dangwal, and M. Qureshi, "ADAPT: Mitigating idling errors in qubits via adaptive dynamical decoupling," in *Proc. MICRO-54: 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2021, pp. 950–962, doi: 10.1145/3466752.3480059.

[15] S. Endo, S. C. Benjamin, and Y. Li, "Practical quantum error mitigation for near-future applications," *Phys. Rev. X*, vol. 8, no. 3, 2018, Art. no. 031027, doi: 10.1103/PhysRevX.8.031027.

[16] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan, "Hybrid quantum-classical algorithms and quantum error mitigation," *J. Phys. Soc. Jpn.*, vol. 90, no. 3, 2021, Art. no. 032001, doi: 10.7566/JPSJ.90.032001.

[17] S. Ferracin et al., "Efficiently improving the performance of noisy quantum computers," 2022, *arXiv:2201.10672*, doi: 10.48550/arXiv.2201.10672.

[18] Jay M. Gambetta et al., "Characterization of addressability by simultaneous randomized benchmarking," *Phys. Rev. Lett.*, vol. 109, no. 24, 2012, Art. no. 240504, doi: 10.1103/PhysRevLett.109.240504.

[19] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, "Digital zero noise extrapolation for quantum error mitigation," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, 2020, pp. 306–316, doi: 10.1109/QCE49297.2020.00045.

[20] H.-L. Huang, D. Wu, D. Fan, and X. Zhu, "Superconducting quantum computing: A review," *Sci. China Inf. Sci.*, vol. 63, no. 8, pp. 1–32, 2020, doi: 10.1007/s11432-020-2881-9.

[21] IBM, IBM Quantum. [Online]. Available: https://quantum-computing.ibm.com/

[22] P. Jurcevic et al., "Demonstration of quantum volume volume 64 on a superconducting quantum computing system," *Quantum Sci. Technol.*, vol. 6, no. 2, 2021, Art. no. 025020, doi: 10.1088/2058-9565/abe519.

[23] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, "Error mitigation extends the computational reach of a noisy quantum processor," *Nature*, vol. 567, no. 7749, pp. 491–495, 2019, doi: 10.1038/s41586-019-1040-7.

[24] Y. Kim et al., "Scalable error mitigation for noisy quantum circuits produces competitive expectation values," 2021, *arXiv:2108.09197*, doi: 10.48550/arXiv.2108.09197.

[25] Y. Li and S. C. Benjamin, "Efficient variational quantum simulator incorporating active error minimization," *Phys. Rev. X*, vol. 7, no. 2, 2017, Art. no. 021050, doi: 10.1103/PhysRevX.7.021050.

[26] A. Lowe, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, "Unified approach to data-driven quantum error mitigation," *Phys. Rev. Res.*, vol. 3, no. 3, 2021, Art. no. 033098, doi: 10.1103/PhysRevResearch.3.033098.

[27] R. LaRose et al., "Mitiq: A software package for error mitigation on noisy quantum computers," *Quantum*, vol. 6, p. 774, 2022, doi: 10.22331/q-2022-08-11-774.

[28] R. LaRose, A. Mari, V. Russo, D. Strano, and W. J. Zeng, "Error mitigation increases the effective quantum volume of quantum computers," 2022, *arXiv:2203.05489*, doi: 10.48550/arXiv.2203.05489.

[29] E. Magesan, J. M. Gambetta, and J. Emerson, "Robust randomized benchmarking of quantum processes," 2010, *arXiv:1009.3639*, doi: 10.48550/arXiv.1009.3639.

[30] E. Magesan, J. M. Gambetta, and J. Emerson, "Characterizing quantum gates via randomized benchmarking," *Phys. Rev. A*, vol. 85, no. 4, 2012, Art. no. 042311, doi: 10.1103/PhysRevA.85.042311.

[31] D. C. McKay, S. Sheldon, J. A. Smolin, J. M. Chow, and J. M. Gambetta, "Three-qubit randomized benchmarking," *Phys. Rev. Lett.*, vol. 122, no. 20, 2019, Art. no. 200502, doi: 10.1103/PhysRevLett.122.200502.

[32] A. Mari, N. Shammah, and William J. Zeng, "Extending quantum probabilistic error cancellation by noise scaling," *Phys. Rev. A*, vol. 104, no. 5, 2021, Art. no. 052607, doi: 10.1103/PhysRevA.104.052607.

[33] B. Pokharel, N. Anand, B. Fortman, and D. A. Lidar, "Demonstration of fidelity improvement using dynamical decoupling with superconducting qubits," *Phys. Rev. Lett.*, vol. 121, no. 22, 2018, Art. no. 220502, doi: 10.1103/PhysRevLett.121.220502.

[34] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018, doi: 10.22331/q-2018-08-06-79.

[35] T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Measuring the capabilities of quantum computers," *Nature Phys.*, vol. 18, no. 1, pp. 75–79, 2022, doi: 10.1038/s41567-021-01409-7.

[36] T. Proctor, S. Seritan, K. Rudinger, E. Nielsen, R. Blume-Kohout, and K. Young, "Scalable randomized benchmarking of quantum computers using mirror circuits," 2021, *arXiv:2112.09853*, doi: 10.48550/arXiv.2112.09853.

[37] Google AI Quantum et al., "Hartree-Fock on a superconducting qubit quantum computer," *Science*, vol. 369, no. 6507, pp. 1084–1089, 2020, doi: 10.1126/science.abb9811.

[38] R. Sagastizabal et al., "Experimental error mitigation via symmetry verification in a variational quantum Eigensolver," *Phys. Rev. A*, vol. 100, no. 1, 2019, Art. no. 010302, doi: 10.1103/PhysRevA.100.010302.

[39] C. Song, J. Cui, H. Wang, J. Hao, H. Feng, and Y. Li, "Quantum computation with universal error mitigation on a superconducting quantum processor," *Sci. Adv.*, vol. 5, no. 9, 2019, Art. no. eaaw5686, doi: 10.1126/sciadv.aaw5686.

[40] A. Seif, Z.-P. Cian, S. Zhou, S. Chen, and L. Jiang, "Shadow distillation: Quantum error mitigation with classical shadows for near-term quantum processors," *PRX Quantum*, vol. 4, no. 1, Jan. 2023, Art. no. 010303, doi: 10.1103/PRXQuantum.4.010303.

[41] Y. Suzuki, S. Endo, K. Fujii, and Y. Tokunaga, "Quantum error mitigation as a universal error reduction technique: Applications from the NISQ to the fault-tolerant quantum computing eras," *PRX Quantum*, vol. 3, no. 1, 2022, Art. no. 010345, doi: 10.1103/PRXQuantum.3.010345.

[42] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, no. 4, 1995, Art. no. R2493, doi: 10.1103/PhysRevA.52.R2493.

[43] A. A. Saki, A. Katabarwa, S. Resch, and G. Umbrarescu, "Hypothesis testing for error mitigation: How to evaluate error mitigation," 2023, *arXiv:2301.02690*, doi: 10.48550/arXiv.2301.02690.

[44] P. Stano and D. Loss, "Review of performance metrics of spin qubits in gated semiconducting nanostructures," *Nature Rev. Phys.*, vol. 4, no. 10, pp. 672–688, 2022, doi: 10.1038/s42254-022-00484-w.

[45] K. Schultz et al., "Reducing the impact of time-correlated noise on zero-noise extrapolation," 2022, *arXiv:2201.11792*, doi: 10.48550/arXiv.2201.11792.

[46] J. Sevilla and C. J. Riedel, "Forecasting timelines of quantum computing," 2020, *arXiv:2009.05045*, doi: 10.48550/arXiv.2009.05045.

[47] K. N. Smith et al., "Timestitch: Exploiting slack to mitigate decoherence in quantum circuits," *ACM Trans. Quantum Comput.*, vol. 4, no. 1, pp. 1–27, 2022, doi: 10.1145/3548778.

[48] A. M. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, no. 5, 1996, Art. no. 793, doi: 10.1103/PhysRevLett.77.793.

[49] R. Takagi, "Optimal resource cost for error mitigation," *Phys. Rev. Res.*, vol. 3, no. 3, 2021, Art. no. 033178, doi: 10.1103/PhysRevResearch.3.033178.

[50] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Phys. Rev. Lett.*, vol. 119, no. 18, 2017, Art. no. 180509, doi: 10.1103/PhysRevLett.119.180509.

[51] M. Urbanek, B. Nachman, and W. A. de Jong, "Error detection on quantum computers improving the accuracy of chemical calculations," *Phys. Rev. A*, vol. 102, no. 2, 2020, Art. no. 022427, doi: 10.1103/PhysRevA.102.022427.

[52] UnitaryFund, UnitaryFund Research, Jul. 2022. [Online]. Available: https://github.com/unitaryfund/research/

[53] L. Viola, E. Knill, and S. Lloyd, "Dynamical decoupling of open quantum systems," *Phys. Rev. Lett.*, vol. 82, no. 12, 1999, Art. no. 2417, doi: 10.1103/PhysRevLett.82.2417.

[54] L. Viola and S. Lloyd, "Dynamical suppression of decoherence in two-state quantum systems," *Phys. Rev. A*, vol. 58, no. 4, 1998, Art. no. 2733, doi: 10.1103/PhysRevA.58.2733.

[55] C. Vuillot, "Is error detection helpful on IBM 5Q chips?," 2017, *arXiv:1705.08957*, doi: 10.48550/arXiv.1705.08957.

[56] Y. Zhang et al., "Variational quantum Eigensolver with reduced circuit complexity," *npj Quantum Inf.*, vol. 8, no. 1, 2022, Art. no. 96, doi: 10.1038/s41534-022-00599-z.

[57] S. Zhang et al., "Error-mitigated quantum gates exceeding physical fidelities in a trapped-ion system," *Nature Commun.*, vol. 11, no. 1, pp. 1–8, 2020, doi: 10.1038/s41467-020-14376-z.

[58] J. Zhang, A. M. Souza, F. D. Brandao, and D. Suter, "Protected quantum computing: Interleaving gate operations with dynamical decoupling sequences," *Phys. Rev. Lett.*, vol. 112, no. 5, 2014, Art. no. 050502, doi: 10.1103/PhysRevLett.112.050502.