Volumetric Benchmarking of Error Mitigation with Qermit

Cristina Cirstoiu[1,4], Silas Dilkes[1,4], Daniel Mills[1,4], Seyon Sivarajah[1], and Ross Duncan[1,2,3]

[1]Quantinuum, Terrington House, 13-15 Hills Road, Cambridge CB2 1NL, UK
[2]Department of Computer and Information Sciences, University of Strathclyde, 26 Richmond Street, Glasgow G1 1XH, UK
[3]Department of Physics and Astronomy, University College London, Gower Street, London, WC1E 6BT, UK
[4]These authors contributed equally:{cristina.cirstoiu, silas.dilkes, daniel.mills}@quantinuum.com

The detrimental effect of noise accumulates as quantum computers grow in size. In the case where devices are too small or noisy to perform error correction, error mitigation may be used. Error mitigation does not increase the fidelity of quantum states, but instead aims to reduce the approximation error in quantities of concern, such as expectation values of observables. However, it is as yet unclear which circuit types, and devices of which characteristics, benefit most from the use of error mitigation. Here we develop a methodology to assess the performance of quantum error mitigation techniques. Our benchmarks are volumetric in design, and are performed on different superconducting hardware devices. Extensive classical simulations are also used for comparison. We use these benchmarks to identify disconnects between the predicted and practical performance of error mitigation protocols, and to identify the situations in which their use is beneficial. To perform these experiments, and for the benefit of the wider community, we introduce *Qermit* – an open source python package for quantum error mitigation. Qermit supports a wide range of error mitigation methods, is easily extensible and has a modular graph-based software design that facilitates composition of error mitigation protocols and subroutines.

1 Introduction

Noise inhibits the development of quantum processors [1, 2]. Techniques for reducing the level and effects of noise have been proposed, and act at each layer in the quantum computing stack. At the hardware level, noise reduction may be achieved through calibration [3], dynamical decoupling [4], pulse-level optimisation [5], etc. At the highest level of abstraction, fault-tolerant methods for encoding and processing information with logical qubits may be used [6, 7].

Between these two extremes there exists two classes of noise reduction techniques. The first are those that increase process fidelity through operations at compile time. This class includes noise-aware routing and qubit allocation [8, 9], noise tailoring [10], and circuit optimisation [9]. The second are those which do not increase the fidelity of prepared states directly, but instead improve accuracy when measuring quantities of concern, such as expected values of observables. Several such *error mitigation* protocols have been proposed, including Zero-Noise Extrapolation (ZNE) [11–13], Probabilistic Error Cancellation (PEC) [11], Clifford Data Regression (CDR) [14], Virtual Distillation [15], and many others [16–18].

Importantly, both of these classes of noise reduction techniques require less information about the experimental setup than do hardware level approaches. Additionally, neither compile time error reduction nor error mitigation protocols require a significant overhead in the number of qubits, as in the case of error correction [19, 20]. This often comes at the cost of an increase in the number of circuits and shots in the case of error mitigation.

The importance of noise suppression at the hardware and logical levels is clear as both are vital to the development of scalable fault-tolerant quantum processors. Compile time approaches to noise reduction have also been extensively benchmarked [21] and shown to perform strongly. What's unclear is the impact of error mitigation. While proof of principle experiments have shown improved accuracy when using these techniques [11–14], there has been no systematic study of their practical effectiveness and limitations. As such it is unclear which circuit dimensions and types, nor quantum computing devices of which characteristics, are well suited to the use of error mitigation. Indeed, the differing assumptions made about the underlying noise by each error mitigation protocol may manifest as unpredictable practical performance.

We clarify the practical utility of error mitigation by introducing and implementing a methodology for the benchmarking of error mitigation protocols. Our benchmarking experiments compare the accuracy of operator expectation value calculations when a selection of error mitigation protocols are employed, and when they are not.

The design of our benchmarks is inspired by volumetric benchmarks of quantum processors [22], allowing us to estimate the 'volume' of the circuit depth and width where the error mitigation protocols perform well. The particular circuits used are

constructed from a variety of application-motivated circuit classes [21]. This ensures our benchmarks are indicative of practical performance, which we use to identify classes of computations where error mitigation may be used fruitfully. Taking a volumetric approach ensures that results of these benchmarks may be quickly compared, and that a wide range of circuit sizes are covered. In this work, we benchmark CDR and ZNE, but our method is applicable to a broad class of error mitigation protocols.

We conduct our experiment on real quantum computing devices, and using classical simulators. Each error mitigation method makes different assumptions about the underlying noise, and as such it is difficult to compare error mitigation methods using noisy simulations alone without introducing bias. This is particularly true since existing noise models are poor predictors of hardware behaviour beyond a few qubits. Indeed our experiments on real hardware demonstrate reduced performance as compared with noisy classical simulation, across different error mitigation methods.

For the benefit of the community, and in order to conduct these benchmarking experiments, we have developed *Qermit*. Qermit is an open-source python package for the design and automated execution of error mitigation protocols. The error mitigation protocols presently available through Qermit include those explored in our benchmarking experiments, namely ZNE and CDR. Qermit also includes implementations of PEC, error mitigation based on frame randomisation [10], and protocols performing correction through characterisation of State Preparation And Measurement (SPAM) errors. In all cases, several variations of each protocol are provided. Qermit provides a common interface to this selection of error mitigation schemes, simplifying their use. Further, Qermit supports the straightforward construction and combination of error mitigation protocols and sub routines, facilitating quick prototyping of new protocols. By virtue of being implemented using Tket [9], Qermit is platform-agnostic, and so may be used with a wide range of quantum hardware, and in conjunction with several common quantum software development kits.

The remainder of this paper is structured as follows. In Section 2 we give an overview of error mitigation and the particular schemes that we investigate in this work. In Section 3 we introduce Qermit and details of the implementations of CDR and ZNE. In Section 4 we introduce the design of our benchmarks, and the philosophy that motivates them. In Section 5 we give the results of the experiments we have conducted. Finally we conclude in Section 6.

2 Quantum Error Mitigation

Error mitigation protocols typically have many steps in common. In Section 2.1 we describe a general framework for error mitigation of observables which takes into account the practical aspects of their implementation on quantum hardware. Recent works [23, 24] have also exploited these common features to analyse the overhead in sample complexity, and to determine (universal) lower bounds. Our approach focusses on the modularity of error mitigation protocols, also exploited by the design of Qermit, and takes into account all quantum and classical resources required for an error mitigation experiment. In Section 2.2 and Section 2.3 we use this framework to describe ZNE and CDR. In Section 2.4 we describe the noise profile assumptions that justify the design choices made when developing ZNE and CDR.

2.1 Unifying Framework

Consider a target input quantum circuit given by a sequence of gates $U = U_1U_2...U_d$, an initial (pure) state ρ_0, and an observable O.[1] The error mitigation protocols studied here output an estimator $\langle\hat{O}\rangle_{EM}$ of the true expectation value $\langle O\rangle = \mathrm{tr}(U\rho_0U^\dagger O)$. This estimator should reduce the noise bias in the estimation of this quantity on the backend.

To achieve this, error mitigation protocols run the given circuit and/or other quantum circuits on *backends*, such as quantum processing units or classical simulators, which may be noisy or ideal. The outputs from backends are binary strings, called *shots*, which may be combined to produce, for example, expectation values.

In generating the estimator $\langle\hat{O}\rangle_{EM}$, many error mitigation protocols employ the following steps.

Data Collection: The first step consists of a noise characterisation procedure $\mathfrak{N}$ that takes as input the target experiment(s) (U, ρ_0, O), along with a set of resource parameters $\mathcal{R}$. $\mathcal{R}$ can include: the total number of distinct circuits K, shots per each circuit $(n_i)_{i=1}^K$, and allowed qubits; the type and amount of classical simulation used; and the backend q with fixed specifications such as the compilation strategy, architecture, and noise features.

$\mathfrak{N}$ involves (i) a series of sub-processes $\mathfrak{N}_1$, ..., $\mathfrak{N}_K$, each of which modify the input circuit U in a method-specific way and (ii) measurement circuits $\mathfrak{M}_1(O), ..., \mathfrak{M}_M(O)$ with classical estimator function $o_{m,i}(z)$ for $m = 1, ..., M$ and $i = 1, ..., k$ where z labels the measurements outcomes.

The data collection step returns a set of (labelled) complex parameters given by $\mathbf{D} = (D^q_{i,m}, D^c_{i,m})$, indexed by each of the sub-processes, where

$$D^q_{i,m} = \frac{1}{n_i}\sum_{s=1}^{n_i}\sum_z o_{m,i}(z)Z_s(z) \qquad (1)$$

[1] This may also be extended to a class of circuits or a set of observables.

with Z_s an independent identically distributed (i.i.d) indicator random variable over measurement outcomes obtained from evaluating $\mathfrak{M}_m(O)\mathfrak{N}_i(U)(\rho_0)$ on the quantum device q. If required and available, $D^c_{i,m}$ corresponds to the exact classical simulation.

Functional Model: An implicit mapping $\mathfrak{F}$ (or a set thereof) between the output parameters of the noise characterisation step and the (unknown) error mitigated estimator $\langle\hat{O}\rangle_{EM}$ so that

$$\mathfrak{F}(\mathbf{D}, \langle\hat{O}\rangle_{EM}) = 0. \tag{2}$$

The specific form of this function is typically motivated by assumptions on the noise characteristics of the quantum device.

Data Processing: This step, is completely classical and aims to produce an output estimator $\langle\hat{O}\rangle_{EM}$ based on fitting the data $\mathbf{D}$ to the functional model $\mathfrak{F}$. Depending on the particular function this may be a simple summation or a classical optimisation algorithm to determine the coefficients of $\mathfrak{F}$.

All the processes involved in producing $\mathbf{D}$ may be described in the quantum combs formalism, which generalises quantum channels to higher order operations [25]. $\mathfrak{N}$ will generally depend on the type of noise a particular method is aiming to mitigate. For example, a set of the sub-processes and measurements may be independent of U or O, with the aim to produce (partial) tomographic information [11]. The framework also allows for adaptive processes, which is to say that $\mathfrak{N}_i$ may depend on outcomes $D_{1,m}$... $D_{i-1,m}$ for a subset of the measurement circuits m. Typically the Data Collection step will include the identity process, which does not modify the circuit U or observable, and produces a noisy (sample mean) estimator $\langle\hat{O}\rangle_N$ of the expectation value of the observable given resources $\mathcal{R}$.

Note that several factors at each step can influence the performance of an error mitigation technique. An example is the accuracy in the noise modelling assumptions that motivate the choice of functional model, and we discuss this further in the case of ZNE and CDR in Section 2.4. The accuracy of the data collected, influenced by the available shot budget, also impacts the accuracy and variance in the final error mitigated observable expectation approximation. We explore this variance in the case of our experiments in Appendix E.4.

2.2 Zero Noise Extrapolation

The Zero Noise Extrapolation (ZNE) method [11, 12] assumes that noise may be controlled by a parameter λ (or more generally a set of parameters) which can be viewed as a proxy for average gate error rates. Then, for a given quantum circuit, the noisy expectation value of a target observable will be a function depending on λ. One may produce different samples of this function by artificially increasing the noise parameter to different levels $\lambda_1, \lambda_2, ..., \lambda_k$. An extrapolation process then produces an estimate of the expected value for $\lambda = 0$, the ideal case where no physical errors occur.

There are several ways in which one may boost physical errors affecting a quantum circuit. One method involves increasing the duration of pulses involved in producing each gate within the circuit [11]. A second approach, which we review here, is to introduce additional gates to obtain a higher depth but equivalent unitary circuit [13].

Data Collection: ZNE involves a series of sub-processes $\mathfrak{N}_{\lambda_1}, ..., \mathfrak{N}_{\lambda_k}$, that take the input circuit $U = U_1 U_2 ... U_d$ and produce a modified, or *folded* [13], circuit

$$\mathfrak{N}_{\lambda_i}(U) = U_1(C_1C_1^\dagger)^{\alpha_1^i} U_2(C_2C_2^\dagger)^{\alpha_2^i} ... U_d(C_dC_d^\dagger)^{\alpha_d^i}. \tag{3}$$

Here $C_iC_i^\dagger = \mathbb{I}$ so as to not introduce logical error, and α_j^i are positive integers. There is flexibility in the choice of the C_i unitaries, and [13] analyse different folding variations: (i) circuit folding when only $\alpha_d^i \neq 0$ and $C_d = U_1...U_d$, (ii) random gate folding with $C_j = U_j$ and α_j^i chosen uniformly at random for a fixed noise level $\lambda_i = \sum_j (2\alpha_j^i + 1)$.

The output of the first step will be $\mathbf{D} := (D_{\lambda_1}, D_{\lambda_2}, ..., D_{\lambda_k})$ where each entry is a sample mean estimator for the expectation value $\mathrm{tr}(O\mathfrak{N}_{\lambda_i}(U)\rho_0[\mathfrak{N}_{\lambda_i}(U)]^\dagger)$ of the target observable with respect to the modified circuit at each noise level, given the fixed set of resources (i.e number of shots).

Functional Model: In the case of ZNE, there are several possibilities for the data-fitting functional. We outline several below which have been explored in Refs. [11, 13].

a) The *polynomial extrapolation* assumes that the dependency on the noise level parameter λ can be expressed as a truncated Taylor series so that the data is fitted to the function

$$D_\lambda = \langle\hat{O}\rangle_{EM} + \sum_{i=1}^{K} F_i \lambda^i, \tag{4}$$

with negligible higher order terms $O(\lambda^{K+1})$ and unknown (complex) parameters F_i. If the number of data points, or equivalently in our case the number of noise parameters k, is at least $K+1$, the number of unknown parameters, then the extrapolation is well defined. In the special case when $k = K+1$ there exist analytic expressions for the coefficients, and the method is referred to as Richardson extrapolation [16].

b) The *exponential extrapolation* assumes the expected values of observables decay exponentially with the noise level parameter λ and the data can then be fitted to

$$D_\lambda = \langle \hat{O} \rangle_{EM} + F(e^{-f\lambda} - 1). \tag{5}$$

c) The *poly-exponential extrapolation* assumes that the exponential decay with the noise level has a polynomial expansion so it is fitted to the function

$$D_\lambda = \langle \hat{O} \rangle_{EM} + F(e^{-\sum_{i=1}^{K} F_i \lambda^i} - 1) \tag{6}$$

2.3 Clifford Data Regression

Clifford Data Regression (CDR) [14] is a machine learning approach to error mitigation. The method relies on the idea that circuits containing a number of T gates logarithmic in the number of qubits can be efficiently simulated [26].

Data Collection: CDR involves a series of subprocesses $\mathfrak{N}_0(U) = U$ and $\mathfrak{N}_1, ..., \mathfrak{N}_k$, each of which modifies the input U, synthesized into a universal Clifford + T gate set, to produce a circuit where all except a small number N_{nc} of T gates are replaced by a single-qubit Clifford gate $\{\mathbb{1}, \mathrm{S}, \mathrm{S}^\dagger, \mathrm{Z}\}$. The resulting unitary circuits $\mathfrak{N}_i(U)$ are efficiently simulated classically and so measurements of O will involve both quantum and classical evaluation. The output will be $\mathbf{D} = (D_0^q, (D_1^q, D_1^c), ..., (D_k^q, \tilde{D}_k^c))$ where each pair consists of the ideal classically simulated expectation value of the target observable O with respect to the modified circuit $D_i^c = \mathrm{tr}(O\mathfrak{N}_i(U)\rho_0[\mathfrak{N}_i(U)]^\dagger)$ and respectively D_i^q a corresponding sample mean estimator evaluated on quantum device with resources $\mathcal{R}$.

Functional Model: The functional that relates an error mitigated estimate $\langle \hat{O} \rangle_{EM}$ of the ideal expected value $\langle O \rangle = \mathrm{tr}(OU\rho_0 U^\dagger)$ to the data obtained in the noise characterisation is

$$\langle \hat{O} \rangle_{EM} = f(D_0^q), \tag{7}$$

where $f = \min_{g \in \mathcal{F}} ||\mathbf{D}^c - g(\mathbf{D}^q)||_2^2$ minimises the distance measure $||\mathbf{D}^c - g(\mathbf{D}^q)||_2^2 = \sum_{i=1}^{k} [D_i^c - g(D_i^q)]^2$ over all (invertible) functions g in a specified class $\mathcal{F}$.

In particular, in [14] the class of fitting functions $\mathcal{F}$ was assumed to be linear so that $f(x) := F_1 x + F_0$ and (assuming $F_1 \neq 0$)

$$\langle \hat{O} \rangle_{EM} = F_1 D_0^q + F_0. \tag{8}$$

2.4 Noise Profile Assumptions

The use of different fitting functions in ZNE and CDR are motivated by an incoherent, Markovian noise model. Such a model is described by the quantum channel $\mathcal{N} = (1-\lambda)\mathcal{I} + \lambda\mathcal{E}$ with $\mathcal{I}$ the identity operation, and $\mathcal{E}$ an arbitrary process. Therefore, the noisy implementation of the target unitary channel $\mathcal{U}(\cdot) := U(\cdot)U^\dagger$ is given by $\tilde{\mathcal{U}} := \mathcal{U}_1 \circ \mathcal{N} \circ \mathcal{U}_2 \circ \mathcal{N} ... \circ \mathcal{U}_d \circ \mathcal{N}$. We can expand this out in terms of linear combinations of channels with coefficients depending on the noise parameter λ to get $\tilde{\mathcal{U}} = (1-\lambda)^d \mathcal{U} + \lambda(1-\lambda)^{d-1}\binom{d}{1}\mathcal{E}^{(1)} + ... + \lambda^d \mathcal{E}^{(d)}$, where the notation $\mathcal{E}^{(k)}$ is an average over all processes in the expansion of $\tilde{\mathcal{U}}$ that have exactly k errors $\mathcal{E}$.

In total there are $\binom{d}{k}$ different ways k errors occur within the circuit. Therefore, $\tilde{\mathcal{U}}$ and the corresponding noisy expectation value can be expressed in terms of a power series in λ with degree at most d, thus motivating the polynomial fit. Alternatively, the analysis in [27] considers approximating the binomial coefficients in the expansion of $\mathcal{U}$ with a Poisson distribution so that $\lambda^k(1-\lambda)^{d-k}\binom{d}{k} \approx e^{-d\lambda}\frac{(d\lambda)^k}{k!}$ and therefore, this noise model with the assumption that $\lambda = O(1/d)$ makes the approximation $\tilde{\mathcal{U}} \approx e^{-d\lambda}[\mathcal{U} + \sum_{k=1}^{d} \frac{(d\lambda)^k}{k!}\bar{\mathcal{E}}^{(k)}]$ valid. The noisy expectation value will then take a similar form which motivates the use of exponential and poly-exponential fitting functions. In particular, for a global depolarising noise model (where $\mathcal{E}$ outputs a fixed state ψ_0) the noisy and exact expected values have a linear relationship

$$\langle O \rangle_N = (1-\lambda)^d \langle O \rangle + (1 - (1-\lambda)^d)\mathrm{tr}(O\psi_0). \tag{9}$$

In such a case, the linear fit in CDR and polynomial fit in ZNE (with $K \leq d$) are not susceptible to noise bias so the true expectation can be recovered exactly, up to finite sampling size errors.

3 Qermit Implementation Details

Qermit has a graph based architecture design, which splits the execution of error mitigation methods into atomic functional processes. These sub-processes may be the submission of a circuit to a QPU, the modification of a circuit for the purposes of error mitigation, or the fitting of models. A graph, with these processes as vertices, is specified to describe how the processes should exchange inputs and outputs.

Qermit is complementary to Mitiq [28], which is an open-source Python toolkit that implements an overlapping set of error mitigation schemes. Qermit takes a different approach that breaks-down the implementation of each protocol into standalone modular units, with a graph describing how each module interacts defined separately. This allows the modules to be easily re-used, modified and composed.

This software architecture allows for vertices to be amended to adapt the protocol where necessary, and relieves developers of some of the work of piping together processes. A modular design additionally means sub-graphs and graphs may be reused in other original error mitigation protocols, which is particularly useful as error mitigation schemes typically have several steps in common. As a result it is ensured

that Qermit: is easily extensible; allows one to readily combine error mitigation protocols; and facilitates quick prototyping of new protocols through the reuse of existing sub-protocols. Combining error mitigation protocols has been shown to be a fruitful endeavour [27, 29, 30], and one which is simplified in Qermit.

3.1 MitRes and MitEx

There are two types of error mitigation methods in Qermit: `MitRes` methods, which modify the distribution of shots retrieved from a backend; and `MitEx` methods, which return a modified expectation value estimator of some observable. In general a `MitRes` or `MitEx` object may perform any modification matching this form, or no modification at all. The instances of `MitRes` and `MitEx` objects in Qermit are designed to perform error mitigation.

`MitRes` and `MitEx` objects are constructed as dataflow graphs, called a `TaskGraph`. Each node of a `TaskGraph` is a `MitTask` object; itself a function that computes some step or sub-process of an error mitigation protocol. Edges of the `TaskGraph` move data between `MitTask` objects which depend on each other. When the `run` function is called on a `MitRes` or `MitEx` object, a topological sort is applied to the graph to order these tasks sequentially. The tasks are then run locally in this sequential order.

In its default construction, as displayed in Fig. 1, a `MitRes` object will simply run each circuit through a backend and gather the results. Similarly, the default construction of a `MitEx` object, as displayed in Fig. 2, will simply estimate the expectation of each desired observable without applying any mitigation methods. To do so it will modify the circuit, appropriately rotating the measurement basis according to the target observable. An example of such a use of the default `MitRes` object can be seen in Appendix A.1, and of the default `MitEx` object in Appendix A.2.1.

3.2 ZNE in Qermit

Several options exist in Qermit for both noise scaling and extrapolation to the zero noise limit. Each noise scaling operation available is of the digital variety, and includes: circuit folding, random gate folding, and odd gate folding. Extrapolation functions available include: exponential, linear, poly-exponential, polynomial, and Richardson. Those chosen for our experiments are discussed in Section 5.

An example `TaskGraph` which corresponds to a ZNE `MitEx` can be seen in Fig. 3. One notices in particular that the initial circuit is duplicated by the `Duplicate MitTask`, before each duplicate is passed to a `iFoldMitEx MitTask`. Each `iFoldMitEx` increases the noise in the circuit by a factor of i, and runs the resulting circuit. The original circuit is also passed through a default `MitEx`, as displayed in

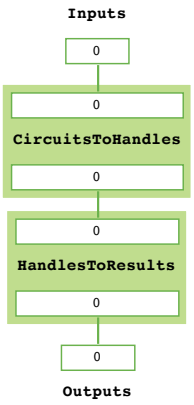


Figure 1: `MitRes TaskGraph`. The `CircuitsToHandles MitTask` takes circuits and a number of shots as input. It ensures the inputted circuits adhere to the requirements of the backend, then submits the circuits to the backend; returning identifying results handles. `HandlesToResults` uses the inputted results handles to retrieve results from the backend, retuning them as outputs.

Fig. 2. The results are collated and used to produce an error mitigated expectation value by the `CollateZNEResults MitTask`. An example of a use of ZNE within Qermit can be seen in Appendix A.2.2.

3.3 CDR in Qermit

One approach to generating the near Clifford training circuit set required for CDR uses Markov chain Monte Carlo techniques [14]. Here a first training circuit is produced, where all but a fixed number of non-Clifford gates are randomly replaced with nearest Clifford gates according to a weighted distribution. Subsequent steps generate circuits sequentially by randomly replacing n_{pairs} of the Clifford gates in the previously generated circuit with their original non-Clifford, and similarly n_{pairs} non-Clifford gates with (nearest) Clifford gates. At each step the new training circuit is accepted/rejected according to a (usually problem-dependent) user-defined maximal likelihood function. A second option, implemented in Qermit, and used in the experiments of Section 5 replaces uniformly at random n_{pairs} of non-Clifford/Clifford pairs in the first training circuit, without producing a chained training set.

Parameters specifying an instance of a CDR `MitEx` include the device backend, the classical simulator backend, the number of non-Clifford gates in the training circuits, the number of pair replacements, and the total number of training circuits. An example `TaskGraph` which corresponds to a CDR `MitEx` can be seen in Fig. 4. The initial circuit is transformed by `CCLStateCircuits` to prepare it for three different experiments. Respectively these are to: run the original circuit on the given backend, run training circuits with a reduced number of non-Clifford gates on the

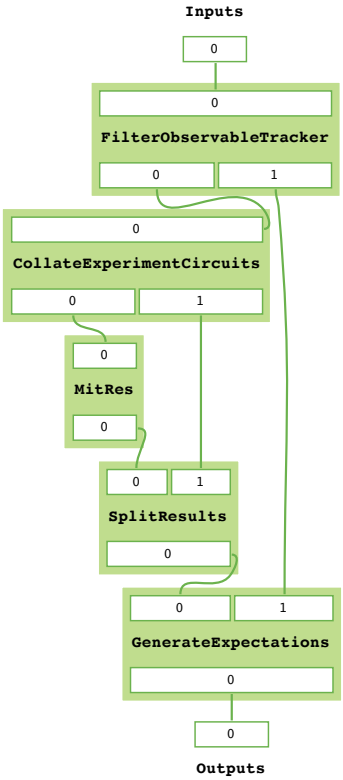


Figure 2: `MitEx TaskGraph`. The `MitRes` task may be expanded as in Fig. 1. Other `MitTask` objects featuring here include: `FilterObservableTracker`, which takes as input a description of the circuit to be implemented, and the observable to be measured, returning circuits modified to perform the measurements necessary to calculate the expectation of the requested observable; `CollateExperimentCircuits`, which reformats the list of circuits to facilitate parallelism; `SplitResults`, which undoes this reformatting; and `GenerateExpectations`, which uses the results to calculate the requested expectation values.

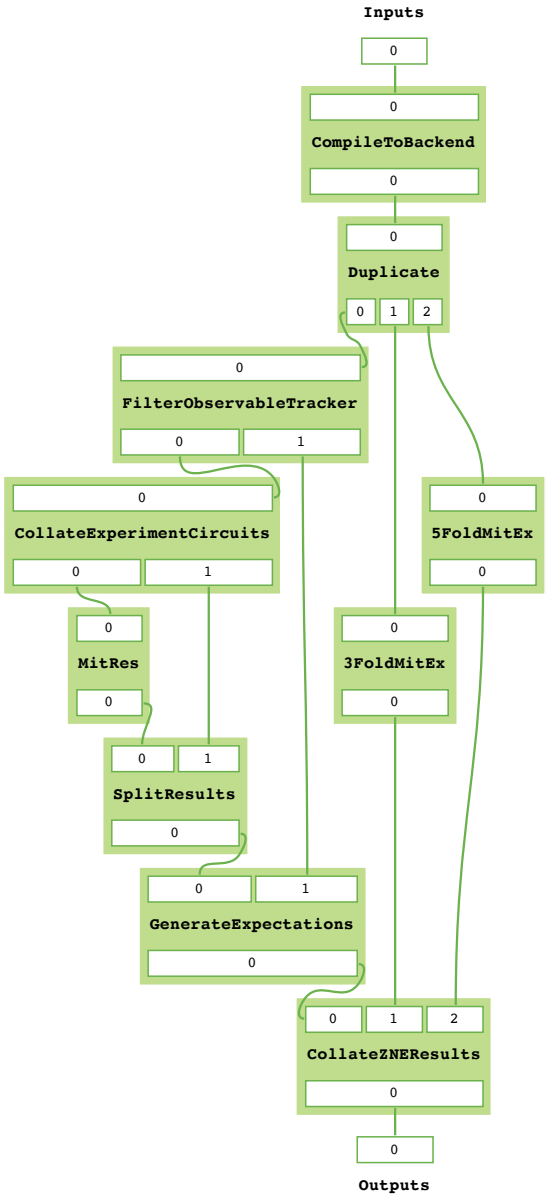


Figure 3: **ZNE** `TaskGraph`. This `TaskGraph` includes: `CompileToBacked`, which ensures the circuit obeys the connectivity and gate set restraints of the backend used; `Duplicate`, which created copies of the inputted circuit; `iFoldMitEx`, which increases the noise in the circuit by a factor i, and runs the resulting circuit using a `MitEx` of Fig. 2; and `CollateZNEResults`, which gathers the experiment results and extrapolates to the zero noise limit. Note that other `MitTask` objects appearing in this figure have parallels in Fig. 2. Indeed the `iFoldMitEx` contains a MitEx `TaskGraph`, which we do not expand for succinctness, as well as a `MitTask` perfoing the noise scaling.

same given backend, and run the training circuits on an ideal classical simulator. Once these experiments have been conducted, a series of checks conducted in `QualityCheckCorrect` ensure that the training data gives a well-conditioned (least-squares) optimisation. `CDRCorrect` finds the fit parameters and produces an error mitigated estimator. An example of the use of CDR can be found in Appendix A.2.3.

4 Benchmark Design

Here we develop a benchmark procedure which can be used to compare error mitigation methods on different backends. To assess the performance of an error mitigation protocol, we introduce the relative error of mitigation in Section 4.1. We further discuss in Section 4.2 factors that influence such performance. Our benchmarks are volumetric by design, with the required circuit structure, presentation style and results interpretation discussed in Section 4.3. Finally we detail the precise circuits we use in Section 4.4.

4.1 Accuracy Measures for Error Mitigation

In order to assess the performance of an error mitigation strategy we propose the following criteria that such a performance metric should satisfy (i) faithful

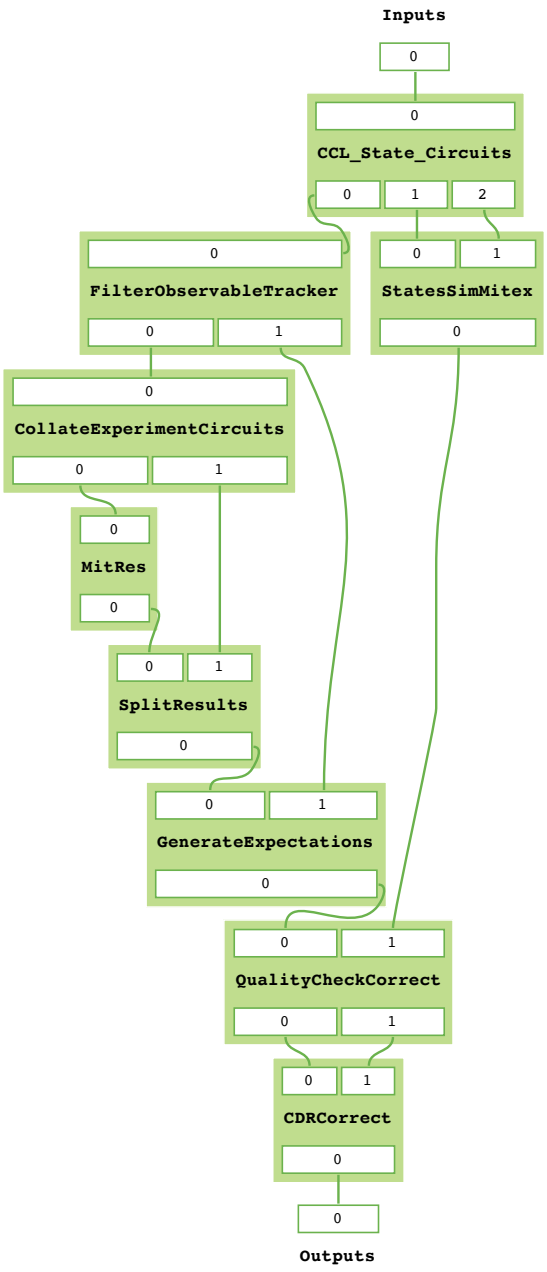


Figure 4: **CDR `TaskGraph`.** Besides those `MitTask` objects which are common to Fig. 2, this `TaskGraph` additionally includes: `CCLStateCircuits`, which outputs the original circuits as output 0, and training circuits copied as outputs 1 and 2; `StatSimMitEx`, which runs these two sets of circuits through a given backend and an ideal classical simulator, returning the results; `QualityCheckCorrect`, which assesses the expected quality of the resulting function model; and `CDRCorrect`, which uses the results from the original and training circuits to find a function model and produce an error mitigated result. Note that `StateSimMitEx` would expand to reveal a `MitEx` for the inputted backend, and a `MitEx` for an ideal classical simulator.

– takes a fixed (zero) value when the error mitigated estimator matches the exact value (ii) operational – relates to improvement in a computational task (iii) efficiently estimable from experimental data and classical processing – allows for scalability.

Definition 4.1. *Let O be an observable with ideal expectation $\langle O\rangle$ with respect to the target state ψ, $\langle\hat{O}\rangle_N$ a sample mean estimator for the corresponding noisy state ρ and $\langle\hat{O}\rangle_{EM}$ the sample mean estimator for the expectation value after applying the error mitigation method. The* absolute error *and* absolute error of mitigation *are respectively*

$$\epsilon_N = |\langle\hat{O}\rangle_N - \langle O\rangle| \tag{10}$$

$$\epsilon_{EM} = |\langle\hat{O}\rangle_{EM} - \langle O\rangle| \tag{11}$$

The relative error of mitigation *is defined as*

$$\epsilon(O) = \frac{|\langle\hat{O}\rangle_{EM} - \langle O\rangle|}{|\langle\hat{O}\rangle_N - \langle O\rangle|}. \tag{12}$$

The intuition for the above definition is that we want a measure that expresses how much closer to the true value is the mitigated estimator compared to the noisy expectation. Note that the relative error of mitigation has the following operational properties (i) faithful $\epsilon = 0$ iff corrected expectation values matches the exact value and (ii) whenever $\epsilon \leq 1$ the mitigation of errors was successful.

Generally, the measures in Definition 4.1 involve a classical simulation computing the true expectation value, and therefore are limited to determine performance of error mitigation schemes in these regimes. In Appendix E.1 we discuss the problem of certifying error mitigation for target states or circuit classes for which there is no available classical simulation. We introduce several performance metrics that satisfy (i) - (iii) and are based on mirroring, or efficiently simulable circuits. For example, mirrored circuits give scalable benchmarks since the exact expectation values depend only on the input (product) state and observable itself; no classical simulation of the circuits is needed in this case. We make use of this approach to benchmarking in Section 5. However, we leave for future work to determine how well these measures predict the general performance of error mitigation on quantum hardware, particularly in regimes approaching quantum advantage.

Furthermore, since both $\langle\hat{O}\rangle_{EM}$ and $\langle\hat{O}\rangle_N$ are estimators they incur a variance due to finite sampling statistics. In general the ratio of two random variables does not have a well defined variance and the ratio of two normally distributed variables with zero mean gives rise to the Cauchy distribution, which is typically heavy tailed. However, under mild conditions one can show [31] that ϵ can be approximated

by normal distribution with mean $\mu = \frac{\mu_{EM}}{\mu_N}$ and variance

$$\sigma_\epsilon^2 = \frac{\sigma_{EM}^2}{(\mu_N - \langle O \rangle)^2} + \frac{\sigma_N^2 \, (\mu_{EM} - \langle O \rangle)^2}{(\mu_N - \langle O \rangle)^4} \tag{13}$$

where σ_{EM}^2 and σ_N^2 are the variance of estimators $\langle \hat{O} \rangle_{EM}$ and $\langle \hat{O} \rangle_N$ due to finite sampling statistics and $\mu_{EM} = \mathbb{E}\langle \hat{O} \rangle_{EM}$, $\mu_N = \mathbb{E}\langle \hat{O} \rangle_N$ correspond to the means in the limit of an infinite number of samples. We discuss in Appendix E.2 the conditions under which the above approximation holds.

4.2 Performance of Error Mitigation Methods

There are a series of factors that can affect the performance of error mitigation methods. Firstly, finite sampling effects are amplified in the error mitigated estimator which incurs a higher variance than the noisy expectation value estimator for a fixed number of shots. This limitation has previously been discussed in Ref. [16] and recent work [23] derives theoretical lower bounds on the sample complexity overhead of error mitigation in terms of the maximal noise bias and distinguishability measures. In particular, for a local depolarising noise model the number of shots required to produce a mitigated estimator with the same accuracy as the noisy estimator scales exponentially with the circuit depth [23, 32]. Practically, if the architecture has a restricted topology then this scaling can even depend exponentially on the number of qubits for sparser graphs that require an $O(n)$ routing overhead [33].

Secondly, the functional model used to produce the error mitigated estimator will generally not fully capture the underlying backend noise effects. This is particularly restrictive for real hardware, where $\langle \hat{O} \rangle_{EM}$ will therefore be susceptible to noise bias.

Thirdly, fitting parameters to the functional model involves a classical optimisation that may be ill conditioned or unstable, partly due to the increased variance in the finite sampling or in the functional fit.

Finally there are several specific regimes where the above issues can be more detrimental to the performance of error mitigation strategies. For example if low levels of noise occur then $\langle \hat{O} \rangle_N$ already produces a good estimator of $\langle O \rangle$ with high accuracy and due to the additional sampling overhead error mitigation will not, on average, improve upon that estimator for a fixed shot budget. Typically error mitigation strategies involve reconstructing the surface defined by $\mathfrak{F}$ from the noisy samples $\mathbf{D}$ in the data collection step – however, if the finite sample error dominates then solutions to the fit parameters will be unstable. This situation can occur for high levels of noise and is typically exacerbated when the target observables have low expected values. In the following section and Appendix E.2 we make these observations more precise.

4.2.1 Amplification of finite sampling variance in the error mitigated estimator

Recall that the error mitigated estimator $\langle \hat{O} \rangle_{EM}$ is produced by classical post-processing which fits the experimental results to a functional model $\mathfrak{F}(\mathbf{D}, \langle \hat{O} \rangle_{EM}) = 0$. The terms $D_{i,m}^q$ are sample mean estimators that will introduce finite sampling effects. In the simplest case one might have a linear functional where $\langle \hat{O} \rangle_{EM} = \sum_{i,m} F_{i,m} D_{i,m}^q$ for some (real) coefficients $F_{i,m}$ and where $D_{i,m}^q$ are defined in Eq. (1). The variance due to finite sampling in the error mitigated estimator is then

$$\sigma_{EM}^2 := Var[\hat{O}_{EM}] = \sum_{m,i} F_{i,m}^2 Var[D_{i,m}^q] \tag{14}$$

$$= \sum_{m,i} \frac{F_{i,m}^2 o_{m,i}}{n_i} \sigma_i^2 \tag{15}$$

where $o_{m,i} := \sum_z o_{m,i}(z)^2$ are constants predetermined from the target observables (and any modified observables required in the data collection step) and recall that n_i are the number of independent samples $\{Z_s\}_{s=1}^{n_i}$ each with variance σ_i^2.

4.3 Volumetric Benchmarking

Volumetric benchmarking of error mitigation assesses the overall performance of a method on a specific backend with a fixed set of total resources $\mathcal{R}$. We employ circuit classes with increasing depth d (as determined by the number of layers of primitive circuits) and qubit number n (which is to say the number of qubits the circuit acts on). Our methodology is inspired by volumetric benchmarks of quantum processors [22] and consists of:

1. Select a class of circuits $\mathcal{C}(n, d)$ and a probability distribution or method to sample C individual circuits.
2. Select a (Pauli) observable O (or set thereof) with fixed locality.
3. Determine relative error of mitigation $\epsilon_i(O)$ for each circuit $\mathcal{C}(n, d)$ labelled by $i \in \{1, ..., C\}$.
4. Determine the median relative error of mitigation over the sampled circuits

$$\bar{\epsilon} = \text{med}_{i=1}^{C}[\epsilon_i(O, n, d)].$$

The choice of circuit classes and sampling methods are flexible and we discuss them in detail in the following section. In our benchmarks we will consider global observables, acting non-trivially on each qubit. Measurements of global observables are typically affected by all errors occurring throughout a circuit whereas a local observable is affected by errors within its light-cone. Indeed, if the noisy operations outside

the light-cone are described by completely positive trace-preserving maps, then their action on the identity observable cancels out. This assumption may fail if, for example, correlated errors across the cone partition occur. Therefore the choice of a global observable will place stronger constraints on depths/qubit number at which error mitigation gives improved estimators. One might also consider observables with a fixed locality constraint determined by applications of interest.

There are two motivations behind the use of medians as a measure of overall success. First, we discussed how finite sampling statistics can give a long-tailed distribution of the relative error of mitigation. Secondly we observe for multiple experiments a similar long-tailed behaviour for the distribution over different circuits in the same class $\mathcal{C}(n, d)$.

4.4 Circuits

In this section, we formally define and motivate the circuits used in benchmarking error mitigation methods. In particular, for each circuit class we define a primitive *layer* type, providing pseudocode generating it. A layer acts on n qubits, where n is called the *qubit number* of the circuit. We specify the depth of a layer, and discuss how layers are composed to create a *depth* d circuit from the given class. This terminology corresponds to that used in Section 4.3. While the qubit number and depth are indicative of the circuit sizes, the exact number of gates will vary between circuit classes. The compiled circuits will additionally depend on the architecture of the backend. This is discussed and displayed in Fig. 5. The particular implementation generating the circuits used in our experiments is available as specified in the Code Availability.

The circuit classes that we make use of are *Random Circuits*, introduced in Section 4.4.1 and inspired by those used for quantum volume experiments [21, 34], and Pauli-Gadget Circuits, introduced in Section 4.4.2 and inspired by ansatz circuits for NISQ algorithms [35, 36]. Collectively, this selection of circuit classes encompass several important potential applications of quantum computing, covering circuits of varied depth, connectivity, and gate types, strengthening the predictive power of our benchmarks.

While these circuit classes allow us to draw conclusions about the performance of error mitigation on specific applications, there are many circuits that would not be covered by such classes. However, the mirroring method discussed in Section 4.4.3, and the experiment design in Section 4.4.4 are sufficiently general to allow for the use of different classes of circuits, as may be desirable to others. General purpose benchmark suites may provide inspiration for such choices [37].

We avoid favouring any device in particular by the design of the benchmark circuits used here. The connectivity and the gate-set assumed by the circuits is very general to ensure we benchmark the error mitigation schemes as general purpose schemes. A compilation step will be required to execute these circuits on a real device. The compilation strategy used may also influence the performance of the error mitigation scheme. While the same compilation strategy is used during error mitigated runs as with noisy runs, compilation passes may have different effects on error mitigated and unaltered circuits. As such we fix the compilation scheme used during our benchmarks to avoid this dependency. Note in particular that caution should be taken in the case of mirrored circuits (see Section 4.4.3) to ensure that the circuits are not compiled to the identity.

4.4.1 Random Circuits

While circuits required for applications are typically not random, sampling from the output distributions of random circuits built from two-qubit gates has been suggested as a means to demonstrate quantum computational supremacy [38–41]. By utilising uniformly random two-qubit unitaries and all-to-all connectivity, *Random Circuits*, introduced now, provide a comprehensive benchmark.

The circuits used here – taken from Ref. [21] and similar to those used for the quantum volume benchmark [34] – are generated according to Algorithm 1, and are illustrated in Fig. 6. This circuit class consist of d *layers* of two-qubit gates acting between a bipartition of the qubits.

By utilising uniformly random two-qubit unitaries, Random Circuits test the ability of the error mitigation scheme to mitigate errors acting on a universal gate set. Allowing two-qubit gates to act between any pair of qubits in the uncompiled circuit, means Random Circuits avoid favouring any device in particular. This choice adheres closely to our motivations of being hardware-agnostic. Because of this generality Random Circuits are complementary to the other classes introduced in this work.

4.4.2 Pauli-Gadget Circuits

Pauli gadgets [43] are quantum circuits implementing an operation corresponding to exponentiating a Pauli tensor. Sequences of Pauli gadgets acting on qubits form *product formula* circuits, most commonly used in Hamiltonian simulation [44]. Many algorithms employing these circuits require fault-tolerant devices, but they are also the basis of trial state preparation circuits in many variational algorithms, which are the most promising applications of noisy quantum computers.

A notable example of this in quantum chemistry is the physically-motivated *Unitary Coupled Cluster*

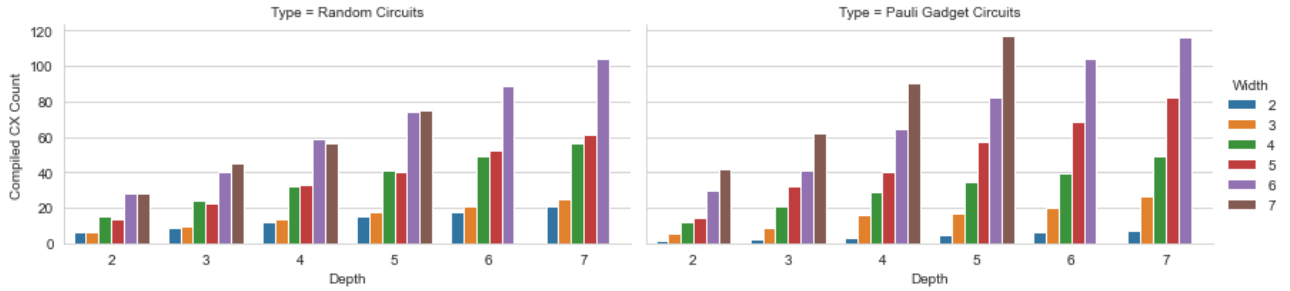


Figure 5: **CX gate count of circuits used in the experiments of Section 5.** These circuits are not mirrored. Compilation is onto the `ibm_lagos` device, as discussed in Appendix B. Bars give the mean CX gate count over the circuits.

Algorithm 1 The pattern for building Random Circuits.

Input: Width, $n \in \mathbb{Z}$, depth, $d \in \mathbb{Z}$ and mirrored $\in \{\text{True}, \text{False}\}$
Output: Circuit, C_n

Initialise n qubits, labelled $q_1, ..., q_n$, to $|0\rangle$

if mirrored **then**
  $d = \lfloor \frac{d}{2} \rfloor$
end if
for each layer t up to depth d **do**
    ▷ This loop's content constitutes a *layer*.

  Divide the qubits into $\lfloor \frac{n}{2} \rfloor$ pairs $\{q_{i,1}, q_{i,2}\}$ at random.

  for all $i \in \mathbb{Z}$, $0 \leq i \leq \lfloor \frac{n}{2} \rfloor$ **do**

    Generate $\mathcal{U}_{i,t} \in \mathrm{SU}(4)$ uniformly at random according to the Haar measure.
    Enact the gate corresponding to the unitary $\mathcal{U}_{i,t}$ on qubits $q_{i,1}$ and $q_{i,2}$.
      ▷ Decompositions of this gate can be found in [42]
    if mirrored **then**
      Enact the gate corresponding to the unitary $\mathcal{U}_{i,t}^{\dagger}$ on qubits $q_{i,1}$ and $q_{i,2}$.
    end if

  end for

end for

Measure all qubits in the computational basis.

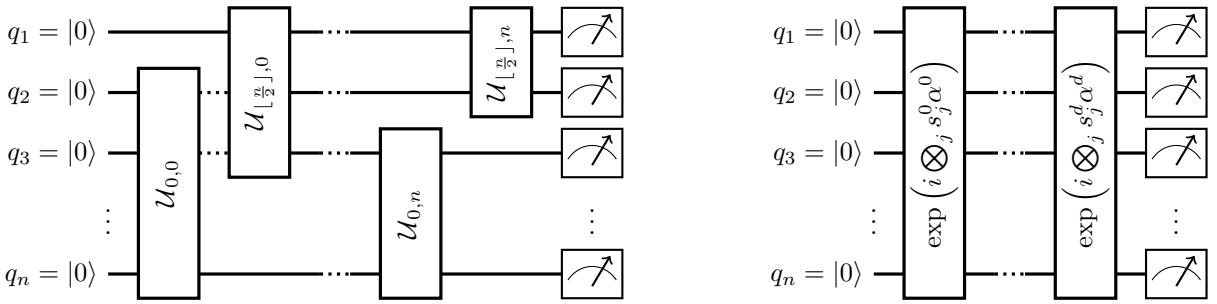
family of trial states used in the variational quantum eigensolver (VQE) [35, 36]. Motivated by these practical applications, we include in our benchmarks circuits with a similar structure. The Pauli-Gadget Circuits are built as in Algorithm 2, and may be visualised as in Fig. 6. They are constructed from several layers of Pauli Gadgets, each acting on a random subset of n qubits. Note that the circuits in this class differ from running end-to-end VQE. Focusing on the state preparation portion of a VQE circuit, we might deduce performance of error mitigation when running the VQE on ansatze of similar type.

4.4.3 Mirrored Circuits

Besides the division by circuit class, given by the specific layer type, circuits are also defined as being mirrored or un-mirrored. While un-mirrored circuits consist of independently generated random layers of a fixed type, mirrored circuits consist of randomly generated layers, each followed by their inverse. Mirrored circuits correspond to an identity operation and thus have the property that their expectation values can be efficiently computed with increasing qubit number and depth.

Note that Mirrored circuits in this work are inspired by, and are a special case of, those introduced in [45]. "Mirroring" in [45] consists of concatenating a circuit with its quasi-inverse (its inverse up to Pauli operations) and inserting Pauli operations before, after and between the circuit and its quasi-inverse. The ideal behaviour of such a circuit would be to produce the input state, inspired by the Loschmidt echo. In our case we take the quasi inverse to be the inverse, and do not add layers of Pauli operations.

In both Algorithm 1 and Algorithm 2 the option to generate mirrored circuits is provided as an input parameter. This has the effect of inverting each layer of the circuit once the layer has been applied. Random layers are generated and inverted a total of $\lfloor \frac{d}{2} \rfloor$ times to ensure that the final depth is d, assuming that $d \in 2\mathbb{Z}$. This results in a circuit implementing the identity, with ideal expectation value that depends only on the observable and input state. Therefore, mirrored circuits can provide scalable benchmarks to



(a) **An example of Random Circuits, as generated by Algorithm 1.**

(b) **An example of Pauli-Gadget Circuits, as generated by Algorithm 2.**

Figure 6: **Example circuits.** These particular examples have mirrored set to false. In the case of mirrored Pauli-Gadget Circuits, one can imagine appending the gate $\exp\left(-i \bigotimes_j s_j^t \alpha^t\right)$ after the gate $\exp\left(i \bigotimes_j s_j^t \alpha^t\right)$. Respectively in the case of mirrored Random Circuits, one can imagine appending the gate $\mathcal{U}_{i,t}^\dagger$ after the gate $\mathcal{U}_{i,t}$. Note that the gates $\mathcal{U}_{i,t}$ in the random circuit class act between 2 randomly selected qubits.

Algorithm 2 The pattern for building Pauli-Gadget Circuits.

Input: Width, $n \in \mathbb{Z}$, depth, $d \in \mathbb{Z}$ and mirrored $\in \{\text{True}, \text{False}\}$
Output: Circuit, C_n

Initialise n qubits, labelled $q_1, ..., q_n$, to $|0\rangle$.

if mirrored **then**
  $d = \lfloor \frac{d}{2} \rfloor$
end if
for each layer t up to depth d **do**
  ▷ This loop's content constitutes a *layer*.

  Select a random string $s^t \in \{\mathbb{1}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}\}^n$
  Generate random angle $\alpha^t \in [0, 2\pi]$
  Enact the Pauli gadget corresponding to the unitary $\exp\left(i \bigotimes_j s_j^t \alpha^t\right)$ on qubits $q_1, ..., q_n$.
  ▷ Decompositions of this gate can be found in [43]
  if mirrored **then**
    Enact the Pauli gadget corresponding to the unitary $\exp\left(-i \bigotimes_j s_j^t \alpha^t\right)$ on qubits $q_1, ..., q_n$.
  end if

end for

Measure all qubits in the computational basis

assess the performance of error mitigation strategies. We discuss this further in Appendix E.1.

4.4.4 Experiment Design

In our experiments, for a given qubit number and depth we choose a fixed number of random instances from each class of Random and Pauli-Gadget circuits and their mirrored versions. The measurements are in the computational basis with the target observables set to $O = \bigotimes_{i=0}^n \mathsf{Z}_i$ and input state $\rho_0 = |0\rangle^{\otimes n}$. The choice of a global observable was motivated in Section 4.3.

From the randomly generated circuit instances we select those with ideal expectation values within a fixed range (specifically we select values from 0.4 to 0.6). As the circuit size increases the probability of generating circuits with expectation values in this range falls exponentially. This limits the size of such circuits that we can generate, demonstrating an additional advantage of the mirrored circuits which in this case have a fixed ideal expectation value 1.

We require this control of the ideal expectation value for the following reason: Random circuits show anti-concentration properties with increasing depth, meaning that for Pauli observables the expectation values will be close to zero with high probability[46]. However, our experiments, particularly the hardware runs, are limited by finite size sampling so that the obtained data $\mathbf{D} = \{D_{i,m}^q\}_{i,m}$ incurs a variance depending on the fixed number of shots. Fitting this data to the surface $\mathfrak{F}(\mathbf{D}, \langle \hat{O} \rangle_{EM}) = 0$ to determine an error mitigated estimator can be numerically unstable if the variance due to finite sampling is comparable to the value of the noisy data $\mathbf{D}$ itself. This is particularly relevant for very low ideal expected values since accumulation of noise within the circuit results in a further decreased value (typically dropping exponentially with number of noisy gates). To avoid this situation, where the error mitigation performance decrease

with increasing depth can be (partly) attributed to the random generation of benchmark circuits, we restrict to high/fixed range expected values. We refer to Appendix E.4 for further details.

5 Results

We use the benchmarks introduced in Section 4 to assess the performance of ZNE and CDR on both superconducting hardware and noisy simulator backends. In Fig. 7 and Fig. 8 we present volumetric plots displaying the relative error of mitigation for Pauli-Gadget Circuits and Random Circuits respectively, when run on both `ibm_lagos` and `ibmq_casablanca`. Extensive noisy simulations are provided in the Appendix D.

5.1 Noisy Classical Simulations

The first set of benchmarks employ a local depolarising noise model with error rate of $e_1 = 10^{-3}$ and $e_2 = 10^{-2}$ for single and two qubit gates respectively.[2] These values are chosen to be broadly inline with those reported by several hardware providers, but do not correspond to any device in particular. In this case there is no restriction on the pairs of qubits between which two qubit gates can act. As such only minimal compilation to the simulator's gate set is required.

The results of these experiments are presented in the volumetric plots of Figs. 11 and 12. For each fixed width/depth (i.e square) 10 circuits are sampled from the specified class, with 5×10^5 shots per circuit for each error mitigation protocol, and 10^5 in the case of the noisy expected value. This shot count is higher than could be taken from real backends at our disposal in a reasonable runtime. This high shot count is used here to present an idealised scenario and illustrate boundaries of qubit count and circuit depth beyond which error mitigation fails to improve the noisy estimator of the expected value. In particular, such boundaries align with the theoretical analysis limiting the depth to scale with a constant inverse of the error rate $d = O(1/e_2)$ [24, 27].

In the case of ZNE, we use an exponential fit and the target circuit is folded repeatedly for a number of $\lambda = 1, 3, 5, 7, 9$ times. The shot budget is distributed equally between these noise scaling values. Curve fitting is achieved via a least-square optimisation. For CDR, we distribute the total shot budget equally to each of the 21 required circuits; namely 20 near-Clifford circuits, and 1 unaltered circuit. Generation of the training set fixes the number of non-Clifford operations to 10 and uniformly at random generates the new near-Clifford circuits by pair replacements from an initial seed circuit. In the case where the total number of non-Clifford gates is less than 10, at least one non-Clifford gate is replaced at random. Additionally, using the classical simulation of these circuits we test that the training data gives a well-conditioned matrix for the linear regression to be performed and generate new training circuits if that is not the case.

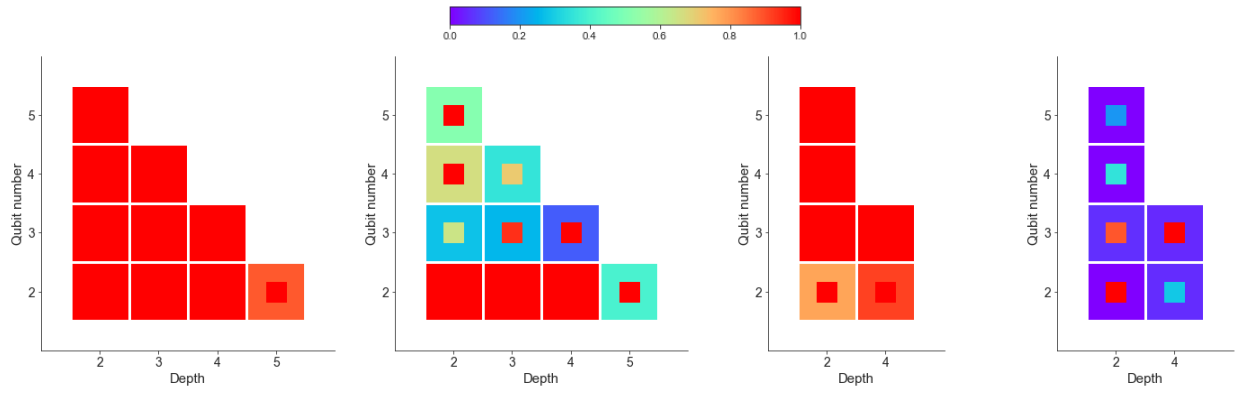
There are circuit sizes where ZNE and CDR are consistently beneficial Comparing first the performance of CDR and ZNE, we note from Figs. 11a and 11b that for Random Circuits in the size ranges explored, both show an improvement over the case where error mitigation is not used. ZNE appears to perform particularly well in this domain, outperforming CDR for each circuit size explored. To identify the limits of the utility of CDR and ZNE we extend the depth of the circuit explored using mirrored circuits, as discussed in Section 4.4. We see from Figs. 11c and 11d that while ZNE outperforms CDR for smaller circuit sizes, the range of circuit sizes where CDR outperforms the use of no error mitigation appears to be larger than for ZNE.

CDR outperforms ZNE with Pauli-Gadget Circuits Comparing Fig. 12 to Fig. 11 we note that while ZNE outperform CDR in case of Random Circuits, this appears not to be the case for Pauli-Gadget Circuits. As shown in Fig. 5, the CX gate count of circuits with a fixed qubit number and depth are similar between the Random Circuits and Pauli-Gadget Circuits classes. As such this improved relative performance of CDR is likely because there are a smaller proportion of non-Clifford gates in Pauli-Gadget Circuits than Random Circuits. In the case where there are few non-Clifford circuits, the training circuits are little different from the original circuit, which improved the accuracy of the CDR training procedure.

More refined device modelling reduces performance The second set of benchmarks employ device emulation that includes many properties of the corresponding real backend used in Section 5.2.[3] These properties include connectivity of the architecture, available gate set, and an expansion of the noise model to include thermal relaxation and readout errors, in addition to a local depolarising model with parameters corresponding to calibration results. Exact values of these properties can be found in Appendix B. The parameters used for ZNE and CDR are identical in this case to those used when performing depolarising noise simulation, with the exception that the total

[2] To create this model we make use of the qiskit noise model module, as described at https://qiskit.org/documentation/apidoc/aer_noise.html. For more details see Code Availability.

[3] To achieve device emulation use the pytket IBMQEmulatorBackend as described at https://cqcl.github.io/pytket-extensions/api/qiskit/api.html. For more details see Code Availability.

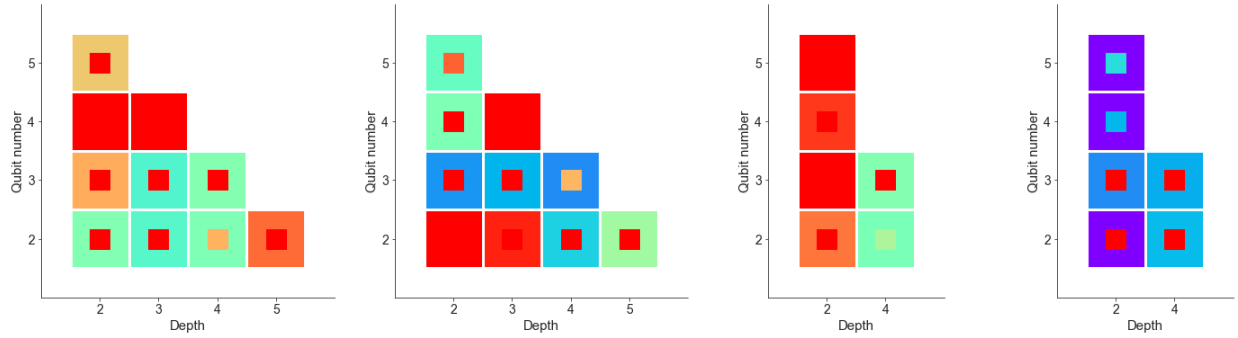


(a) `ibm_lagos`, ZNE, Pauli-Gadget Circuits.

(b) `ibm_lagos`, CDR, Pauli-Gadget Circuits.

(c) `ibm_lagos`, ZNE, Mirrored Pauli-Gadget Circuits.

(d) `ibm_lagos`, CDR, Mirrored Pauli-Gadget Circuits.



(e) `ibmq_casablanca`, ZNE, Pauli-Gadget Circuits.

(f) `ibmq_casablanca`, CDR, Pauli-Gadget Circuits.

(g) `ibmq_casablanca`, ZNE, Mirrored Pauli-Gadget Circuits.

(h) `ibmq_casablanca`, CDR, Mirrored Pauli-Gadget Circuits.

Figure 7: **Volumetric plots for performance of error mitigation on Pauli-Gadget Circuits.** `ibm_lagos` and `ibmq_casablanca` are used with ZNE or CDR on Pauli-Gadget Circuits and mirrored Pauli-Gadget Circuits. Median $\bar{\epsilon}$ (outer square) and worst-case (inner square) relative error of mitigation over the sampled circuits are shown, metric that ranges between $(0, 1)$ for successful mitigation with lower values corresponding to better performance. Each square corresponds to 5 sampled circuits of the specified type with a total 1.6×10^5 total shot budget per each mitigated expectation value.

shot budget is reduced to 1.6×10^5 for the mitigated experiment, and 0.32×10^5 for the unmitigated experiments. The shot budget is reduced to match that of the real device experiments, where computing resources are more scarce. Comparing Figs. 11a, 11c, 12a and 12c to Figs. 13a, 13c, 14a and 14c we see a marked fall in the performance of ZNE when additional device constraints are impose by emulation.

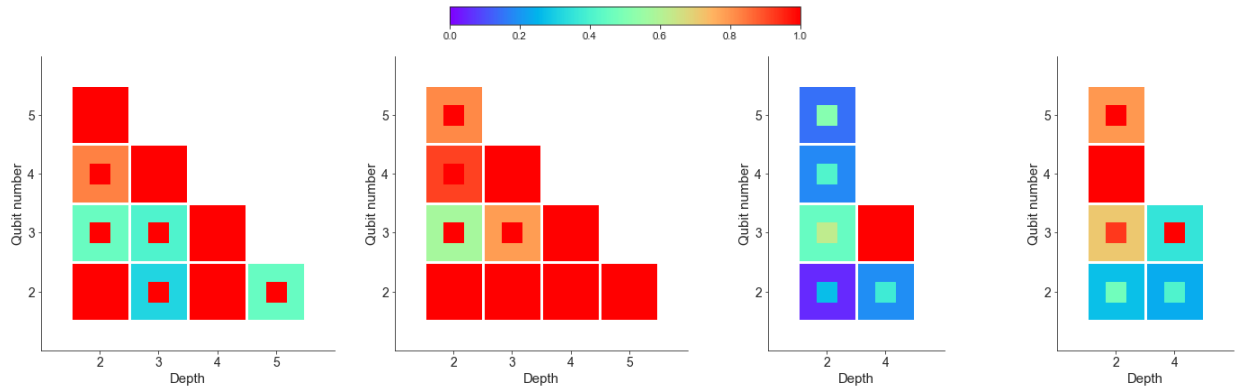
5.2 Quantum Hardware Backend

Finally, we perform volumetric benchmarks to assess and compare the performance of ZNE and CDR error mitigation on `ibmq_lagos` and `ibmq_casablanca` as measured by the relative error of mitigation.

The experiments used in the volumetric benchmarks are produced according to Section 4.4 and employ the two types of circuit classes as described in Section 4.4.4. For each fixed qubit number and depth we use 5 circuits on which we sequentially perform CDR, ZNE and no error mitigation. The same sets of circuits are used on both `ibmq_casablanca` and `ibm_lagos`. As discussed, the parameters used for ZNE and CDR are the same as in the case of the depolarising noise simulations, and the device emulation experiments. As in the case of the emulation experiments, the shot budget is set to 1.6×10^5 for the error mitigated experiments, and 0.32×10^5 for the unmitigated experiments.

It is important to emphasize that for any given circuit we run the two mitigation schemes, and the unmitigated experiments, back-to-back so that any time drift in the device's noise parameters do not skew the performance comparison. For both devices the calibration data during these experiments can be found in Table 2 and Table 1. Furthermore, they share the same architecture connectivity.

Improvements are inconsistent As seen in Figs. 7 and 8, for a fixed qubit number and depth (as measured by the number of layers from a specific circuit class), we typically found at least one sampled circuit for which error mitigation does not improve upon the noisy expectation value. This occurred for both error mitigation methods even when their me-



(a) `ibm_lagos`, ZNE, random.

(b) `ibm_lagos`, CDR, Random Circuits.

(c) `ibm_lagos`, ZNE, Mirrored Random Circuits.

(d) `ibm_lagos`, CDR, Mirrored Random Circuits.

(e) `ibmq_casablanca`, ZNE, Random Circuits.

(f) `ibmq_casablanca`, CDR, Random Circuits.

(g) `ibmq_casablanca`, ZNE, Mirrored Random Circuits.

(h) `ibmq_casablanca`, CDR, Mirrored Random Circuits.

Figure 8: **Volumetric plots for performance of error mitigation on Random Circuits.** `ibm_lagos` and `ibmq_casablanca` are used with ZNE or CDR on Random Circuits and mirrored Random Circuits. Median $\bar{\epsilon}$ (outer square) and worst-case (inner square) relative error of mitigation over the sampled circuits are shown, metric that ranges between $(0, 1)$ for successful mitigation with lower values corresponding to better performance. Each square corresponds to 5 sampled circuits of the specified type with a total 1.6×10^5 total shot budget per each mitigated expectation value.

dian performance gave a 2-fold or higher error reduction (i.e $\bar{\epsilon} \leq 0.5$). This stands in contrast with the emulation results, where there was a significant size region with different (n, d) for which error mitigation was successful even in the worst case. Method parameters and shot budget were the same for emulation and device experiments.

Improvements are less on `ibm_lagos` Strikingly, whilst the dominating two qubit error rates are within 10% for the two devices, performance of error mitigation drastically decreased on `ibm_lagos`. In particular, as seen in Figs. 7a and 7c, the structured Pauli-Gadget Circuits showed little to no improvement over the noisy expectation value when exponential ZNE was performed. This is not due to `ibm_lagos` producing results with low absolute error. As indicated in Tables 3 and 4 the absolute error of the results from `ibm_lagos` are in fact typically higher than that of `ibmq_casablanca`. Comparatively, as seen in Fig. 7e, on `ibmq_casablanca` for small qubit sizes $n = 3$ and 2, ZNE mitigation was successful with a median relative error of 0.51 respectively 0.50 for 4 Pauli circuit layers. However, comparing Figs. 7f and 7h against Figs. 7b and 7d, we see that the performance of CDR was consistent between the two devices for this circuit class, and on average scaled well to the largest circuit sizes considered.

On the other hand, we see from Fig. 8 that for Random circuits both ZNE and CDR had worse performance on `ibm_lagos` compared to `ibmq_casablanca`, with the limits $\bar{\epsilon} > 1$ reached within the sizes considered in both depth and qubit number. For this circuit class, as seen in Figs. 8e and 8g, the performance of ZNE on `ibm_casablanca` consistently reached on average a reduction in the noisy estimator's absolute error by 2 or more for all sizes considered. This was generally slightly better than CDR, although both methods still improved the noisy estimators even for larger qubit sizes and depths.

As a result of these features, each appearing unique to the particular device, we do not expect the results presented here to carry over generically to other devices. This is particularly the case for architectures other than superconducting.

Mirrored circuits are optimistic In the case of mirrored circuits, the performance was more consistent between the two devices. Notably, as seen in Figs. 7d and 7h, on mirrored Pauli gadget circuits CDR decreased, on average, the absolute error in the noisy estimator by up to an order of magnitude for most circuit sizes. This corresponds in Fig. 7 to a median relative error of mitigation of 0.1 or less, and the results were similar for both devices. Comparing Figs. 7a and 7e against Figs. 7c and 7g shows ZNE mirrored Pauli circuits were a good predictor of the poor performance of non-mirrored circuits with similar dimensions.

Surprisingly, the relative error of mitigation for mirrored random circuits improved from 0.45 on `ibmq_casablanca` to 0.15 on `ibm_lagos` for $n = 5$ qubits, as seen in Fig. 8g and Fig. 8c, respectively. This suggests that while mirrored circuits provide scalable benchmarks to assess performance of error mitigation, they generally overestimate the median relative error for circuits of similar sizes using both of the methods considered. As such they may be taken to provide an prediction of an upper bound on the performance of error mitigation schemes, and so indicating when a scheme should be expected to bring no benefit.

6 Conclusions and Outlook

In this work we introduce Qermit – an open source package for error mitigation with a composable software architecture that allows for combining different error mitigation methods and facilitates development of new techniques.

A key question we address is *given a particular device and class of circuits, which error mitigation has, on average, a better performance?* To that aim, we design volumetric benchmarks to assess the performance of error mitigation methods for a given class of quantum circuits with varying qubit number and depth. The methodology is based on frameworks for benchmarking quantum devices [22]. We use this framework here to delineate the boundary beyond which a given error mitigation technique fails on average to give an improved estimator of the target expectation value.

In this setup, we compare the performance of ZNE and CDR implementations on current superconducting hardware. We show how noise in real devices places even stronger constraints on the scalability of error mitigation than theoretical lower bounds as for example derived in [23] or determined from classical noisy simulations [29]. Qualitatively, we found that even for a low total shot budget ($\approx 10^5$ per mitigated expected value) CDR generally outperforms ZNE on structured Pauli circuits and slightly underperforms on random SU(4) circuits. This generic behaviour is captured by the emulation which includes depolarisation, thermal relaxation and readout in the noise model as well as the device's architecture. However, emulation largely overestimates the performance with successful mitigation for significantly larger circuit sizes compared to those accessible with real hardware. In contrast, a simplified depolarising noise model without connectivity constraints fails to predict even qualitative features of error mitigation performance.

On real hardware, we find that for circuits of the same type and depth the relative error of mitigation can vary extensively. Often, it results in unsuccessful mitigation in the worst case even when on average the error in the expectation value is reduced by an order of magnitude. We run the same benchmarks on two superconducting devices with the same connectivity and comparable calibration data and find a high level of variability in the performance of both error mitigation methods. This was particularly striking for ZNE, where the use of digitised method to artificially increase noise level may be more susceptible to the different noise profiles. These results flag two issues; first, that fine grained noise characterisation is needed to predict behaviour of error mitigation on hardware and second, that without access to the ideal expectation value one needs to validate that error mitigation has succeeded.

Our benchmarks include mirrored circuits, which can be efficiently classically simulated as the number of qubits grows. As such, mirrored circuits provide a useful benchmark of the performance of error mitigation, particularly when classical simulation of their un-mirrored counterparts is inaccessible. We observe that, given a fixed qubit number, depth, shot budget and device, mirrored circuits typically produced a lower average relative error of mitigation for both ZNE and CDR compared to un-mirrored circuits from the same class. This illustrates that mirrored circuits give an upper bound on the relative error of mitigation. This feature remains constant across devices, circuit classes and schemes, and so it gives a way to infer when EM will not produce improved results.

Due to the emphasis on composability, Qermit is particularly well suited to explore the application of multiple noise tailoring and mitigation techniques for the same circuit and observable. In future work we will explore the performance of combined methods, which is particularly well motivated by recent results that show significant improvements when using Pauli frame randomisation, ZNE and dynamical decoupling in combination, and with additional bespoke device calibration [47, 48]. In particular, for these combined methods the error mitigated estimates of magnetisation gave a better approximation than the leading approximate classical tensor network simulations for 2D Ising models on up to $N = 127$ qubits. Note that these results do not contradict those of this paper as we do not explore here combinations of error miti-

gation schemes, or techniques which make such extensive use of device characteristics. However, this indicates that Qermit's composable design makes it suitable for testing how useful error mitigation can be to achieving practical quantum advantage.

It would also be pertinent to understand how the performance of error mitigation schemes varies between different device architectures, such as ion traps devices. As we have seen, for example when comparing the classical simulations of Appendix D.2 to those performed on a real device in Section 5.2, idealised noise models can be a poor predictor of real performance. This discrepancy will likely be emphasised by using different devices, between which noise models vary greatly. Even apparently similar devices were shown in Section 5.2 to perform differently, and so we do not expect our results to generically carry over to other superconducting devices. Qermit facilitates such experiments as it is implemented with pytket [9], which gives access to an assortment of devices. While in Appendix D.2 we note some observations regarding the performance of error mitigation with changing noise levels, a more refined and extensive study with more complex noise models would be of great interest.

Since all parameters including total shot budget per mitigated expected value are fixed, the discrepancy in performance between the two devices can be attributed to the noise bias and different noise profiles. Therefore, it would be of interest to determine how the results of the volumetric benchmarks change for the *same* device at different calibration times. This was not possible in the case of `ibm_casablanca` as the machine was taken out of use during our experiments, so we leave this type of assessment for future work.

It will also be particularly interesting to explore the effect of finite sampling in the case of ion traps, where the rate at which shots are generated is lower. Such restrictions will impact the accuracy with which a functional model, as discussed in Section 2.1, can be approximated. This could manifest, for example, in the accuracy of the learnt extrapolation function in the case of ZNE, and so the variance in the learnt zero noise limit. In Appendix E.4 we introduce theoretical tools to explore this particular point, inspired by numerical results revealing strong performance variation depending on shot budget availability [29]. We look to future work to establish strictly how the relative error of mitigation varies not just with circuit size as explored here, but also with shot count.

Finally, would be insightful to compare our approach, targeting an understanding of the practical utility of error mitigation, to those exploring information theoretic bounds to performance [23, 49–51]. In such results, lower bounds on the resources required to ensure beneficial use of error mitigation are derived, and are shown to grow exponentially with circuit size. We anticipate that the approach of our work will be complementary to those results, firstly as a means of verifying the predicted scaling. Secondly we imagine that our techniques could be used to explore more complicated unknown noise models.

Data Availability A complete set of results can be found in [52]. This additionally includes the circuits used, and other data required to reproduce the experiments.

Qermit Installation and Usage Information on the installation and use of Qermit is available in the documentation at:

`www.qerm.it`

and the user manual at:

`https://cqcl.github.io/Qermit/manual/`

Qermit is available for the Mac, Linux and Windows operating systems via PyPI. Qermit may be installed using the command line operation:

```
pip install qermit
```

The source code for Qermit is hosted at:

`https://github.com/CQCL/qermit`

Acknowledgements Thanks to Steven Herbert and Hussain Anwar for insightful feedback. This work was supported by Innovate UK Project No: 10001712. "Noise Analysis and Mitigation for Scalable Quantum Computation". We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

References

[1] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331/q-2018-08-06-79.

[2] Aram W. Harrow and Ashley Montanaro. "Quantum computational supremacy". In: *Nature* 549.7671 (Sept. 2017), pp. 203–209. ISSN: 1476-4687. DOI: 10.1038/nature23458. URL: http://dx.doi.org/10.1038/nature23458.

[3] Sarah Sheldon et al. "Procedure for systematically tuning up cross-talk in the cross-resonance gate". In: *Phys. Rev. A* 93 (6 June 2016), p. 060302. DOI: 10.1103/PhysRevA.93.060302. URL: https://link.aps.org/doi/10.1103/PhysRevA.93.060302.

[4] Lorenza Viola, Emanuel Knill, and Seth Lloyd. "Dynamical Decoupling of Open Quantum Systems". In: *Phys. Rev. Lett.* 82 (12 Mar. 1999), pp. 2417–2421. DOI: 10.1103/PhysRevLett.82.2417. URL: https://link.aps.org/doi/10.1103/PhysRevLett.82.2417.

[5] Andre R. R. Carvalho et al. "Error-Robust Quantum Logic Optimization Using a Cloud Quantum Computer Interface". In: *Phys. Rev. Applied* 15 (6 June 2021), p. 064054. DOI: 10.1103/PhysRevApplied.15.064054. URL: https://link.aps.org/doi/10.1103/PhysRevApplied.15.064054.

[6] Peter W. Shor. "Scheme for reducing decoherence in quantum computer memory". In: *Phys. Rev. A* 52 (4 Oct. 1995), R2493–R2496. DOI: 10.1103/PhysRevA.52.R2493. URL: https://link.aps.org/doi/10.1103/PhysRevA.52.R2493.

[7] Barbara M. Terhal. "Quantum error correction for quantum memories". In: *Rev. Mod. Phys.* 87 (2 Apr. 2015), pp. 307–346. DOI: 10.1103/RevModPhys.87.307. URL: https://link.aps.org/doi/10.1103/RevModPhys.87.307.

[8] Prakash Murali et al. "Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers". In: *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS '19. Providence, RI, USA: Association for Computing Machinery, 2019, pp. 1015–1029. ISBN: 9781450362405. DOI: 10.1145/3297858.3304075. URL: https://doi.org/10.1145/3297858.3304075.

[9] Seyon Sivarajah et al. "t|ket⟩: a retargetable compiler for NISQ devices". In: *Quantum Science and Technology* 6.1 (Nov. 2020), p. 014003. DOI: 10.1088/2058-9565/ab8e92. URL: https://doi.org/10.1088/2058-9565/ab8e92.

[10] Joel J. Wallman and Joseph Emerson. "Noise tailoring for scalable quantum computation via randomized compiling". In: *Phys. Rev. A* 94 (5 Nov. 2016), p. 052325. DOI: 10.1103/PhysRevA.94.052325. URL: https://link.aps.org/doi/10.1103/PhysRevA.94.052325.

[11] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. "Error Mitigation for Short-Depth Quantum Circuits". In: *Phys. Rev. Lett.* 119 (18 Nov. 2017), p. 180509. DOI: 10.1103/PhysRevLett.119.180509. URL: https://link.aps.org/doi/10.1103/PhysRevLett.119.180509.

[12] Ying Li and Simon C. Benjamin. "Efficient Variational Quantum Simulator Incorporating Active Error Minimization". In: *Phys. Rev. X* 7 (2 June 2017), p. 021050. DOI: 10.1103/PhysRevX.7.021050. URL: https://link.aps.org/doi/10.1103/PhysRevX.7.021050.

[13] Tudor Giurgica-Tiron et al. "Digital zero noise extrapolation for quantum error mitigation". In: *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. 2020, pp. 306–316. DOI: 10.1109/QCE49297.2020.00045.

[14] Piotr Czarnik et al. "Error mitigation with Clifford quantum-circuit data". In: *Quantum* 5 (Nov. 2021), p. 592. ISSN: 2521-327X. DOI: 10.22331/q-2021-11-26-592. URL: https://doi.org/10.22331/q-2021-11-26-592.

[15] William J. Huggins et al. "Virtual Distillation for Quantum Error Mitigation". In: *Phys. Rev. X* 11 (4 Nov. 2021), p. 041036. DOI: 10.1103/PhysRevX.11.041036. URL: https://link.aps.org/doi/10.1103/PhysRevX.11.041036.

[16] Suguru Endo et al. "Hybrid Quantum-Classical Algorithms and Quantum Error Mitigation". In: *Journal of the Physical Society of Japan* 90.3 (2021), p. 032001. DOI: 10.7566/JPSJ.90.032001. eprint: https://doi.org/10.7566/JPSJ.90.032001. URL: https://doi.org/10.7566/JPSJ.90.032001.

[17] Kishor Bharti et al. "Noisy intermediate-scale quantum algorithms". In: *Rev. Mod. Phys.* 94 (1 Feb. 2022), p. 015004. DOI: 10.1103/RevModPhys.94.015004. URL: https://link.aps.org/doi/10.1103/RevModPhys.94.015004.

[18] Bálint Koczor. "Exponential Error Suppression for Near-Term Quantum Devices". In: *Phys. Rev. X* 11 (3 Sept. 2021), p. 031057. DOI: 10.1103/PhysRevX.11.031057. URL: https://link.aps.org/doi/10.1103/PhysRevX.11.031057.

[19] Austin G. Fowler et al. "Surface codes: Towards practical large-scale quantum computation". In: *Phys. Rev. A* 86 (3 Sept. 2012), p. 032324. DOI: 10.1103/PhysRevA.86.032324. URL: https://link.aps.org/doi/10.1103/PhysRevA.86.032324.

[20] Joe O'Gorman and Earl T. Campbell. "Quantum computation with realistic magic-state factories". In: *Phys. Rev. A* 95 (3 Mar. 2017), p. 032338. DOI: 10.1103/PhysRevA.95.032338. URL: https://link.aps.org/doi/10.1103/PhysRevA.95.032338.

[21] Daniel Mills et al. "Application-Motivated, Holistic Benchmarking of a Full Quantum Computing Stack". In: *Quantum* 5 (Mar. 2021), p. 415. ISSN: 2521-327X. DOI: 10.22331/q-2021-03-22-415. URL: https://doi.org/10.22331/q-2021-03-22-415.

[22] Robin Blume-Kohout and Kevin C. Young. "A volumetric framework for quantum computer benchmarks". In: *Quantum* 4 (Nov. 2020), p. 362. ISSN: 2521-327X. DOI: 10.22331/q-2020-11-15-362. URL: https://doi.org/10.22331/q-2020-11-15-362.

[23] Ryuji Takagi et al. "Fundamental limits of quantum error mitigation". In: *npj Quantum Information* 8.1 (Sept. 2022), p. 114. ISSN: 2056-6387. DOI: 10.1038/s41534-022-00618-z. URL: https://doi.org/10.1038/s41534-022-00618-z.

[24] Zhenyu Cai. *A Practical Framework for Quantum Error Mitigation*. 2021. arXiv: 2110.05389 [quant-ph].

[25] G. Chiribella, G. M. D'Ariano, and P. Perinotti. "Quantum Circuit Architecture". In: *Phys. Rev. Lett.* 101 (6 Aug. 2008), p. 060401. DOI: 10.1103/PhysRevLett.101.060401. URL: https://link.aps.org/doi/10.1103/PhysRevLett.101.060401.

[26] Sergey Bravyi and David Gosset. "Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates". In: *Phys. Rev. Lett.* 116 (25 June 2016), p. 250501. DOI: 10.1103/PhysRevLett.116.250501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.116.250501.

[27] Zhenyu Cai. "Multi-exponential error extrapolation and combining error mitigation techniques for NISQ applications". In: *npj Quantum Information* 7.1 (May 2021). ISSN: 2056-6387. DOI: 10.1038/s41534-021-00404-3. URL: http://dx.doi.org/10.1038/s41534-021-00404-3.

[28] Ryan LaRose et al. "Mitiq: A software package for error mitigation on noisy quantum computers". In: *Quantum* 6 (Aug. 2022), p. 774. ISSN: 2521-327X. DOI: 10.22331/q-2022-08-11-774. URL: https://doi.org/10.22331/q-2022-08-11-774.

[29] Daniel Bultrini et al. "Unifying and benchmarking state-of-the-art quantum error mitigation techniques". In: *Quantum* 7 (June 2023), p. 1034. ISSN: 2521-327X. DOI: 10.22331/q-2023-06-06-1034. URL: https://doi.org/10.22331/q-2023-06-06-1034.

[30] Andrea Mari, Nathan Shammah, and William J. Zeng. "Extending quantum probabilistic error cancellation by noise scaling". In: *Physical Review A* 104.5 (Oct. 2021). ISSN: 2469-9934. DOI: 10.1103/physreva.104.052607. URL: http://dx.doi.org/10.1103/PhysRevA.104.052607.

[31] Eloísa Díaz-Francés and Francisco J. Rubio. "On the existence of a normal approximation to the distribution of the ratio of two independent normal random variables". In: *Statistical Papers* 54.2 (May 2013), pp. 309–323. ISSN: 1613-9798. DOI: 10.1007/s00362-012-0429-2. URL: https://doi.org/10.1007/s00362-012-0429-2.

[32] Samson Wang et al. *Can Error Mitigation Improve Trainability of Noisy Variational Quantum Algorithms?* 2021. arXiv: 2109.01051 [quant-ph].

[33] Alexander Cowtan et al. "On the Qubit Routing Problem". en. In: (2019). DOI: 10.4230/LIPICS.TQC.2019.5. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10397/.

[34] Andrew W. Cross et al. "Validating quantum computers using randomized model circuits". In: *Phys. Rev. A* 100 (3 Sept. 2019), p. 032328. DOI: 10.1103/PhysRevA.100.032328. URL: https://link.aps.org/doi/10.1103/PhysRevA.100.032328.

[35] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5.1 (July 2014), p. 4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213. URL: https://doi.org/10.1038/ncomms5213.

[36] Panagiotis Kl. Barkoutsos et al. "Quantum algorithms for electronic structure calculations: Particle-hole Hamiltonian and optimized wave-function expansions". In: *Phys. Rev. A* 98 (2 Aug. 2018), p. 022322. DOI: 10.1103/PhysRevA.98.022322. URL: https://link.aps.org/doi/10.1103/PhysRevA.98.022322.

[37] Thomas Lubinski et al. *Application-Oriented Performance Benchmarks for Quantum Computing*. 2023. arXiv: 2110.03137 [quant-ph].

[38] Adam Bouland et al. "On the complexity and verification of quantum random circuit sampling". In: *Nature Physics* 15.2 (Feb. 2019), pp. 159–163. ISSN: 1745-2481. DOI: 10.1038/s41567-018-0318-2. URL: https://doi.org/10.1038/s41567-018-0318-2.

[39] Ramis Movassagh. *Efficient unitary paths and quantum computational supremacy: A proof of average-case hardness of Random Circuit Sampling*. 2018. arXiv: 1810.04681 [quant-ph].

[40] Sergio Boixo et al. "Characterizing quantum supremacy in near-term devices". In: *Nature Physics* 14.6 (June 2018), pp. 595–600. ISSN: 1745-2481. DOI: 10.1038/s41567-018-0124-x. URL: https://doi.org/10.1038/s41567-018-0124-x.

[41] Scott Aaronson and Lijie Chen. *Complexity-Theoretic Foundations of Quantum Supremacy Experiments*. 2016. arXiv: 1612.05903 [quant-ph].

[42] Robert R. Tucci. *An Introduction to Cartan's KAK Decomposition for QC Programmers*. 2005. arXiv: quant-ph/0507171 [quant-ph].

[43] Alexander Cowtan et al. "Phase Gadget Synthesis for Shallow Circuits". In: *Electronic Proceedings in Theoretical Computer Science* 318 (Apr. 2020), pp. 214–229. ISSN: 2075-2180. DOI: 10.4204/eptcs.318.13. URL: http://dx.doi.org/10.4204/EPTCS.318.13.

[44] Dominic W. Berry et al. "Efficient Quantum Algorithms for Simulating Sparse Hamiltonians". In: *Communications in Mathematical Physics* 270.2 (Mar. 2007), pp. 359–371. ISSN: 1432-0916. DOI: 10.1007/s00220-006-0150-x. URL: https://doi.org/10.1007/s00220-006-0150-x.

[45] Timothy Proctor et al. "Measuring the capabilities of quantum computers". In: *Nature Physics* 18.1 (Dec. 2021), pp. 75–79. ISSN: 1745-2481. DOI: 10.1038/s41567-021-01409-7. URL: http://dx.doi.org/10.1038/s41567-021-01409-7.

[46] Alexander M. Dalzell, Nicholas Hunter-Jones, and Fernando G. S. L. Brandão. "Random Quantum Circuits Anticoncentrate in Log Depth". In: *PRX Quantum* 3 (1 Mar. 2022), p. 010333. DOI: 10.1103/PRXQuantum.3.010333. URL: https://link.aps.org/doi/10.1103/PRXQuantum.3.010333.

[47] Youngseok Kim et al. "Evidence for the utility of quantum computing before fault tolerance". In: *Nature* 618.7965 (June 2023), pp. 500–505. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06096-3. URL: https://doi.org/10.1038/s41586-023-06096-3.

[48] Youngseok Kim et al. "Scalable error mitigation for noisy quantum circuits produces competitive expectation values". In: *Nature Physics* 19.5 (May 2023), pp. 752–759. ISSN: 1745-2481. DOI: 10.1038/s41567-022-01914-3. URL: https://doi.org/10.1038/s41567-022-01914-3.

[49] Yihui Quek et al. *Exponentially tighter bounds on limitations of quantum error mitigation*. 2023. arXiv: 2210.11505 [quant-ph].

[50] Ryuji Takagi, Hiroyasu Tajima, and Mile Gu. *Universal sampling lower bounds for quantum error mitigation*. 2022. arXiv: 2208.09178 [quant-ph].

[51] Kento Tsubouchi, Takahiro Sagawa, and Nobuyuki Yoshioka. *Universal cost bound of quantum error mitigation based on quantum estimation theory*. 2023. arXiv: 2208.09385 [quant-ph].

[52] Cristina Cirstoiu et al. *Volumetric Benchmarking of Error Mitigation with Qermit: Experimental Data*. Version 1.0. Zenodo, Apr. 2022. DOI: 10.5281/zenodo.6472281. URL: https://doi.org/10.5281/zenodo.6472281.

A Code Snippets

A.1 MitRes

The `MitRes.run` method takes a list of `CircuitShots` objects as an argument. Each `CircuitShots` objects contains the basic information required to run a circuits, namely; a state preparation circuit, and the number of shots to run for the circuit.

The example code snippet:

```
from qermit import MitRes, CircuitShots
from pytket import Circuit
from pytket.extensions.qiskit import
    AerBackend

mitres = MitRes(backend=AerBackend())
c = Circuit(2, 2).H(0).Rz(0.25, 0).CX(1,
     0).measure_all()
results = mitres.run([CircuitShots(
    Circuit=c, Shots=50)])
print(results[0].get_counts())

graph = mitres.get_task_graph()
```

produces the output:

```
Counter({(0, 0): 30, (1, 0): 20})
```

with `graph` being the `TaskGraph` of Fig. 1.

A.2 MitEx

The `MitEx.run` method takes a list of `ObservableExperiment` objects as an argument. Each `ObservableExperiment` objects contains the basic information required to estimate the expectation value of an observable: a state preparation circuit, a dictionary between symbols and parameter values (where appropriate), a pytket `QubitPauliOperator` detailing the operator being measured and used for preparing measurement circuits, and the number of shots to run for each measurement circuit. The following is an example of an `ObservableExperiment` set up.

```
from qermit import AnsatzCircuit,
    ObservableExperiment,
    ObservableTracker, SymbolsDict
from pytket.pauli import Pauli,
    QubitPauliString
from pytket.utils import
    QubitPauliOperator
from pytket import Circuit, Qubit

qubit_pauli_string = QubitPauliString([
    Qubit(1), Qubit(2)], [Pauli.Z, Pauli.
    Z])
```

```
qubit_pauli_operator =
    QubitPauliOperator({
    qubit_pauli_string: 1.0})
ansatz_circuit = AnsatzCircuit(
    Circuit=Circuit(3).X(0).X(1), Shots
        =50, SymbolsDict=SymbolsDict()
)
experiment = ObservableExperiment(
    AnsatzCircuit=ansatz_circuit,
    ObservableTracker=ObservableTracker(
        qubit_pauli_operator),
)
```

In the following we discuss the definition of a `MitEx` object in several cases. In each we will use the `ObservableExperiment` defined above. In all cases an experiment returns a `QubitPauliOperator` object containing an expectation value for each `QubitPauliString`.

A.2.1 Default `MitEx` Example

In its default version, a `MitEx` object will append a measurement circuit for each `QubitPauliString` to the ansatz circuit and execute it through the pytket `Backend` the `MitEx` object is defined by. In particular the example code snippet:

```
from qermit import MitEx
from pytket.extensions.qiskit import
    IBMQEmulatorBackend

device_backend = IBMQEmulatorBackend('
    ibm_lagos', hub='', group='', project
    ='')
mitex = MitEx(backend=device_backend)
mitex_results = mitex.run([experiment])
print(mitex_results[0])

graph = mitex.get_task_graph()
```

produces the output:

```
{(Zq[1], Zq[2]): -0.920000000000000}
```

with `graph` being the `TaskGraph` of Fig. 2. Note that as we have used a noisy simulator, the returned value is different from the ideal value of -1.

A.2.2 ZNE `MitEx` in Qermit

The ZNE `MitEx` has the required inputs: `backend`, the backend on which the circuits will be run; and `noise_scaling_list`, a list of factors by which the noise is scaled. Optionally the type of folding used can be specified via the `_folding_type` keyword argument, which expects a `Folding` object. The fit used to extrapolate results can be specified via the `_fit_type` keyword argument, which expects a `Fit` object.

The example code snippet:

```
from qermit.zero_noise_extrapolation
    import gen_ZNE_MitEx, Folding, Fit

zne_mitex = gen_ZNE_MitEx(
    backend=device_backend,
    noise_scaling_list=[3, 5],
    folding_type=Folding.circuit,
    fit_type=Fit.exponential,
)

mitex_results = zne_mitex.run([
    experiment])
print(mitex_results[0])

graph = zne_mitex.get_task_graph()
```

produces the output:

```
{(Znode[1], Znode[2]):
    -0.986666667477977}
```

with `graph` being the `TaskGraph` of Fig. 3. Note that ZNE had returned a value closer to the ideal value of -1 than did the example in Appendix A.2.1 where no error mitigation was used.

A.2.3 CDR `MitEx` in Qermit

The CDR `MitEx` has required inputs: `device_backend`, the possibly noisy backend used to approximate the expectation of the inputted circuits; `simulator_backend`, an ideal classical simulator; `n_non_cliffords`, the number of non-Clifford gates in the training circuits; `n_pairs`; and `total_state_circuits`, the total number of training circuits to use.

The example code snippet:

```
from qermit.
    clifford_noise_characterisation
    import gen_CDR_MitEx

cdr_mitex = gen_CDR_MitEx(
    device_backend=device_backend,
    simulator_backend=AerBackend(),
    n_non_cliffords=2,
    n_pairs=2,
    total_state_circuits=50,
)

mitex_results = cdr_mitex.run([
    experiment])
print(mitex_results[0])

graph = cdr_mitex.get_task_graph()
```

produces the output:

```
{(Zq[1], Zq[2]): -1.00000000000000}
```

with `graph` being the `TaskGraph` of Fig. 4. Here we again see an improvement on the result from Appendix A.2.1.

B Backend Properties

The coupling map, describing the connectivity between the qubits, of the `ibm_lagos` and

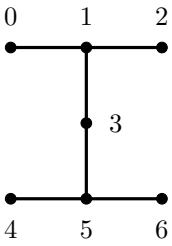


Figure 9: **`ibm_lagos` and `ibmq_casablanca` coupling map.** Vertices in the coupling map correspond to qubits, while edges indicate that 2-qubit gates can be acted between the qubits which correspond to the vertices connected by the edge.

Property	Average Value
T_1 time	128.1
T_2 time	99.50
Single qubit error rate	2.307×10^{-4}
CX error rate	8.973×10^{-3}
Readout error rate	1.164×10^{-2}

Table 1: **`ibm_lagos` device properties.**

`ibmq_casablanca` devices is as seen in Fig. 9. The gates available on the both devices are:

$$\{\mathsf{CX}, \mathsf{RZ}, \mathsf{SX}, \mathsf{X}, \mathbb{1}\} \tag{16}$$

Emulation of the devices requires compilation to the coupling map described in Fig. 9, and a rebasing of the gates used by the circuits to the limited but universal set of Eq. (16).

Average noise properties reported by the devices over the time our experiments were conducted are seen in Tables 1 and 2

C Additional Results Details

Here we present some additional details about the experiments discussed in Section 5. In particular in Table 3 and Table 4 we give the average absolute errors and absolute error of mitigation, as defined in Definition 4.1, for the results presented in Fig. 7 and Fig. 8 respectively. Note that we present mean values here, rather than medians as in Fig. 7 and Fig. 8.

Property	Average Value
T_1 time (μs)	108.5
T_2 time (μs)	124.1
Single qubit error rate	1.713×10^{-4}
CX error rate	8.075×10^{-3}
Readout error rate	2.081×10^{-2}

Table 2: **`ibm_casablanca` device properties.**

D Behaviour of EM Methods Under Noise Models

D.1 Global Depolarising Noise Model

Depolarising noise acts isotropically on quantum states, hence its effect on error mitigation schemes is straightforward to analyse. It takes any state ρ acting on N qubits to

$$\mathcal{D}(\rho) = (1-p)\rho + p\frac{\mathbb{I}}{2^N} \tag{17}$$

where p is the depolarisation factor. It commutes with any other linear map so $\mathcal{D} \circ \mathcal{U} = \mathcal{U} \circ \mathcal{D}$ for any unitary channel $\mathcal{U}$. Suppose that a unitary circuit U contains d_1 single qubit gates and d_2 two-qubit gates, with average error p_1 and p_2 respectively.

If we model the error as globally depolarising, then we have that for any *traceless* observable O the noisy expected value $\langle O \rangle_N$ (in the limit of infinite shots) depends on the exact value $\langle O \rangle$ via

$$\langle O \rangle_N = (1-p_1)^{d_1}(1-p_2)^{d_2}\langle O \rangle \tag{18}$$

For this simplified noise model, the relative error of mitigation remains constant and is independent of the observable and the specific circuit structure. It depends only on the error mitigation method and the number of noisy gates and their average errors.

D.1.1 Zero Noise Extrapolation - Polynomial Fit

We assume that the error rates are increased artificially via unitary folding with noise stretching factors $\alpha_0 = 1$, $\alpha_1 = 2k_1 + 1$ and $\alpha_2 = 2k_2 + 1$. Then the mitigated expected value under a polynomial fit via Richardson extrapolation is

$$\langle O \rangle_{EM} = F_0 \langle P \rangle_{\alpha_0} + F_1 \langle P \rangle_{\alpha_1} + F_2 \langle P \rangle_{\alpha_2} \tag{19}$$

where the coefficients satisfy

$$F_i = \prod_{i \neq j} \frac{\alpha_j}{\alpha_i - \alpha_j} \tag{20}$$

and $\langle O \rangle_{\alpha_i} = \gamma^{\alpha_i} \langle O \rangle$ where $\gamma := (1-p_1)^{d_1}(1-p_2)^{d_2}$. Therefore the relative error mitigation factor is

$$\epsilon_{ZNE,poly} = \frac{1 - F_0\gamma - F_1\gamma^{\alpha_1} - F_2\gamma^{\alpha_2}}{1-\gamma} \tag{21}$$

For example, the minimal number of folds e.g $k_1 = 1$ and $k_2 = 2$ gives $F_0 = \frac{15}{8}$, $F_1 = -\frac{5}{4}$ and $F_2 = \frac{3}{8}$.

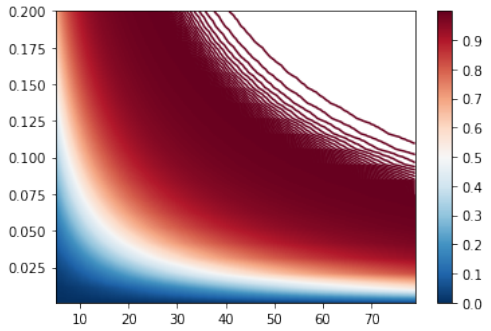
This analytic expression allows us to explore the relationship between the relative error of mitigation and the depolarising factor. Fig. 10a reveals a sharp increase in expected relative error of mitigation as the depolarisation factor grows.

		ibm_lagos						ibmq_casablanca					
		mirrored			un-mirrored			mirrored			un-mirrored		
Width	Depth	none	ZNE	CDR	none	ZNE	CDR	none	ZNE	CDR	none	ZNE	CDR
2	2	0.053	0.046	0.016	0.025	0.426	0.119	0.062	0.052	0.012	0.032	0.016	0.178
2	3				0.026	0.034	0.020				0.035	0.010	0.031
2	4	0.078	0.104	0.005	0.070	0.110	0.098	0.079	0.033	0.052	0.068	0.029	0.047
2	5				0.047	0.049	0.023				0.055	0.049	0.039
3	2	0.189	0.321	0.070	0.073	0.509	0.018	0.163	0.604	0.084	0.063	0.050	0.008
3	3				0.227	1.152	0.051				0.237	0.384	0.232
3	4	0.562	0.721	0.059	0.254	1.017	0.043	0.350	0.286	0.163	0.211	0.403	0.030
4	2	0.508	0.642	0.051	0.253	0.898	0.256	0.307	0.305	0.024	0.252	0.435	0.180
4	3				0.495	1.367	0.180				0.359	1.222	0.455
5	2	0.562	0.595	0.017	0.365	1.209	0.282	0.335	0.604	0.015	0.335	0.427	0.141
Average		0.325	0.404	0.036	0.184	0.677	0.109	0.216	0.314	0.058	0.165	0.302	0.134

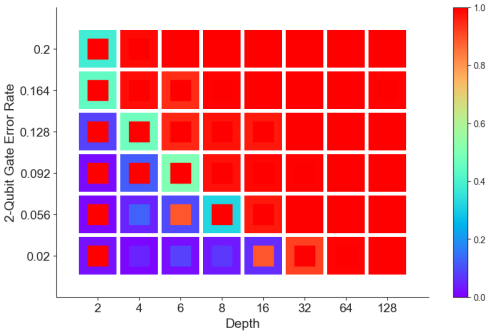
Table 3: **Mean absolute error and absolute error of mitigation for Pauli-Gadget Circuits experiments presented in Fig. 7.**

		ibm_lagos						ibmq_casablanca					
		mirrored			un-mirrored			mirrored			un-mirrored		
Width	Depth	none	ZNE	CDR	none	ZNE	CDR	none	ZNE	CDR	none	ZNE	CDR
2	2	0.076	0.009	0.019	0.048	0.048	0.036	0.089	0.011	0.018	0.062	0.026	0.031
2	3				0.071	0.032	0.090				0.110	0.046	0.037
2	4	0.121	0.028	0.028	0.040	0.053	0.067	0.125	0.011	0.029	0.089	0.028	0.065
2	5				0.063	0.069	0.095				0.112	0.052	0.067
3	2	0.089	0.040	0.051	0.119	0.095	0.136	0.116	0.042	0.035	0.082	0.042	0.057
3	3				0.208	0.092	0.203				0.303	0.248	0.339
3	4	0.328	0.404	0.213	0.183	0.321	0.233	0.252	0.122	0.093	0.112	0.211	0.110
4	2	0.232	0.048	0.203	0.246	0.669	0.285	0.177	0.055	0.270	0.144	0.068	0.185
4	3				0.288	1.100	0.368				0.243	0.080	0.180
5	2	0.244	0.050	0.244	0.223	0.584	0.259	0.227	0.102	0.193	0.224	0.416	0.352
Average		0.182	0.097	0.126	0.199	0.306	0.177	0.164	0.057	0.106	0.148	0.122	0.142

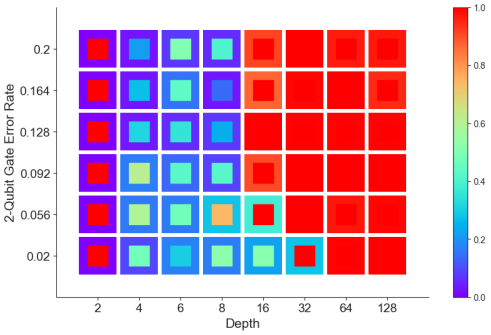
Table 4: **Mean absolute error and absolute error of mitigation for Random Circuits experiments presented in Fig. 8.**



(a) Contour plot of relative error of mitigation in terms of $D = d_1 + d_2$ (x-axix) and error rate $p = p_1 = p_2$ (y-axis) for ZNE with polynomial extrapolation. This is a plot of the function seen in Eq. (21).



(b) Volumetric plot of relative error of mitigation with 4-qubit Pauli-Gadget Circuits depth and 2-qubit depolarising factor. Exponential fit ZNE is used and 500000 shots are taken per circuits. The depolarising factor on 1-qubit gates is a tenth of that on 2-qubit gates.



(c) Volumetric plot of relative error of mitigation with 4-qubit Pauli-Gadget Circuits depth and 2-qubit depolarising factor. Exponential fit CDR is used and 500000 shots are taken per circuits. The depolarising factor on 1-qubit gates is a tenth of that on 2-qubit gates.

Figure 10: **Analytical and numeric calculations of relative error of mitigation with changing depolarising factor.**

D.1.2 Zero Noise Extrapolation - Exponential Fit

The effect of global depolarising noise model is that for different noise levels $\langle O \rangle_{\alpha_i} = \gamma^{\alpha_i} \langle O \rangle$. This means that an exponential ZNE fit exactly captures the underlying noise model with the error mitigated observable given by

$$\langle O \rangle_{EM} = (\langle O \rangle_{\alpha_0})^{\frac{\alpha_1}{\alpha_1 - \alpha_0}} (\langle O \rangle_{\alpha_1})^{\frac{\alpha_0}{\alpha_0 - \alpha_1}} \tag{22}$$

for two levels of noise α_0, α_1. Therefore, in the absence of finite sampling errors $\epsilon_{ZNE,exp} = 0$.

D.1.3 Learning with Clifford Data Regression

Clifford-based learning is specifically designed for (global) stochastic Pauli noise; a case in which the linear ansatz exactly matches the noisy expectation values.

$$\langle O \rangle_N = F_1 \langle O \rangle + F_0 \tag{23}$$

Therefore, in the absence of sampling errors the method should fully correct for the depolarising noise model, and so $\epsilon_{CDR} = 0$.

D.2 Noisy Classical Simulations

We perform extensive classical simulations of the experiments conducted in Section 5, with the results also discussed there. In particular depolarising noise simulations with Random Circuits and Pauli-Gadget Circuits are found in Figs. 11 and 12 respectively.

Besides the fixed depolarising factor simulations presented in Figs. 11 and 12 and discussed in Section 5 we additionally explore simulations with varying noise in Fig. 10. As discussed in Appendix D.1.1 such an analytical study is possible in the case of ZNE with polynomial fit, and is presented in Fig. 10a. By comparison, in other areas of this work we consider exponential extrapolation, which in the idealised case of global depolarising noise results in non zero relative error of mitigation only as a result of finite sample errors. Fig. 10b reveals that for 4 qubit Pauli-Gadget Circuits in the more realistic case of single and two qubit gate local depolarising errors, and with errors due to sampling, there is a sharp increase in the relative error of mitigation as the noise level increases. For the same set of experiments conducted with CDR we again expect 0 relative error of mitigation in the presence of global depolarising errors and in the absence of finite sampling errors. As seen in Fig. 10c CDR is less susceptible to finite sampling errors in the presence of local depolarising noise than is ZNE, speaking to the comparative robustness of the functional model fitting step of the two schemes.

Finally, emulations additionally classically simulates noise, motivated by properties of the device. In the case of the emulates experiments shown here this noise models includes:

Readout error: Single qubit readout errors on measurements.

Single-qubit errors: depolarizing error followed by a thermal relaxation error on each qubit the gate acts on.

Two-qubit gate errors: depolarizing error followed by single-qubit thermal relaxation error on each qubit participating in the gate.

In particular, emulations with Random Circuits and Pauli-Gadget Circuits are found in Fig. 13 and Fig. 14 respectively.

E Performance of Error Mitigation

E.1 Certifying Error Mitigation

In practical applications we don't have a priori access to the exact value $\langle O\rangle$, as this is the quantity we aim to estimate in the first place. Typically proof of principle demonstrations of error mitigation strategies require a classical simulation to compare against, and the measures introduced in Definition 4.1 also assume that $\langle O\rangle$ can be directly computed. Therefore, the problem of assessing the performance of error mitigation on a specific circuit/observable in regimes beyond classical simulations becomes similar to that of certifying quantum computations. For the purpose of benchmarking the techniques for increasing depth and qubit number one may employ scalable benchmarks consisting, for example, of mirrored circuits. Mirrored circuits – with the general form $C(w,d) := U_1U_1^\dagger U_2U_2^\dagger ...U_dU_d^\dagger$ where C_i are sampled from a specified class of primitive circuits. Then for any observable the ideal expectation of the target state $\rho_{\text{mirror}} = C(w,d)\rho_0 C(w,d)^\dagger$ is $\text{tr}(O\rho_{\text{mirror}}) = \text{tr}(O\rho_0)$. For example, if O is a Pauli observable and ρ_0 corresponds to the pure product state $|0\rangle^{\otimes w}$, then the mirrored circuits will give ideal expected values of either $\{0,1,-1\}$, which can be efficiently determined. Then the relative error of mitigation for the mirrored circuit

$$\epsilon_{\text{mirror}}(P) = \frac{|\langle P\rangle_{EM,\tilde{\rho}_{\text{mirror}}} - \langle P\rangle_{\rho_0}|}{|\langle P\rangle_{N,\tilde{\rho}_{\text{mirror}}} - \langle P\rangle_{\rho_0}|} \tag{24}$$

can be used as a proxy for the performance of error mitigation on the target state $\rho = U_1...U_d\,\rho_0\,U_1^\dagger....U_d^\dagger$ and observable P.

E.2 Relative Error of Mitigation Under Finite Sampling Statistics

In [31] the authors derive conditions for when the ratio of two independent normally distributed random variables can itself be approximated by a normal distribution. The following is a direct corollary of that result.

Lemma E.1. *Let X, Y be two independent normal variables with positive means μ_X, μ_Y and variances σ_X^2 and σ_Y^2. Suppose the coefficient of variation $\delta_Y := \frac{\sigma_Y}{\mu_Y} \leq \lambda < 1$. $\forall \epsilon > 0$ there exists a value $\eta(\epsilon)$ such that if $0 < \delta_Y \leq \eta(\epsilon)$ then the probability distribution function F_Z of $Z = X/Y$ approximates a normal distribution G with mean $\mu = \frac{\mu_X}{\mu_Y}$ and variance*

$$\sigma = \frac{\mu_X^2}{\mu_Y^2}\left(\frac{\sigma_X^2}{\mu_X^2} + \frac{\sigma_Y^2}{\mu_Y^2}\right) \tag{25}$$

with $|F_Z(z) - G(z)| \leq \epsilon$ on the interval $z \in [\mu - \frac{\sigma}{\lambda}, \mu + \frac{\sigma}{\lambda}]$.

In our case, $X = \langle\hat{O}\rangle_{EM} - \langle O\rangle$ corresponds to the difference between the error mitigated estimator and the exact value and $Y = \langle\hat{O}\rangle_N - \langle O\rangle$ corresponds to the difference between the noisy estimator and exact value of the target expectation value. The expected values are then $\mu_X = \mathbb{E}\langle O\rangle_{EM} - \langle O\rangle$ respectively $\mu_Y = \mathbb{E}\langle O\rangle_N - \langle O\rangle$ with variances $Var[\langle\hat{O}\rangle_{EM}] = \sigma_{EM}^2$ and $Var[\langle\hat{O}\rangle_N] = \sigma_N^2$.

Since we are interested in relative error of mitigation and this corresponds to absolute values of Z, then one can assume that μ_X and μ_Y are positive quantities because if that happens not to be the case for a fixed observable then we have the freedom to redefine the random variable X and Y so as to ensure the positivity constraint.

In the following section we discuss the remaining condition that controls the signal to noise ratio and how the tension between low levels of noise and high variance in the noisy estimators make it difficult to quantify the performance of error mitigation in this regime.

E.3 Detrimental Effect of Low Noise Level

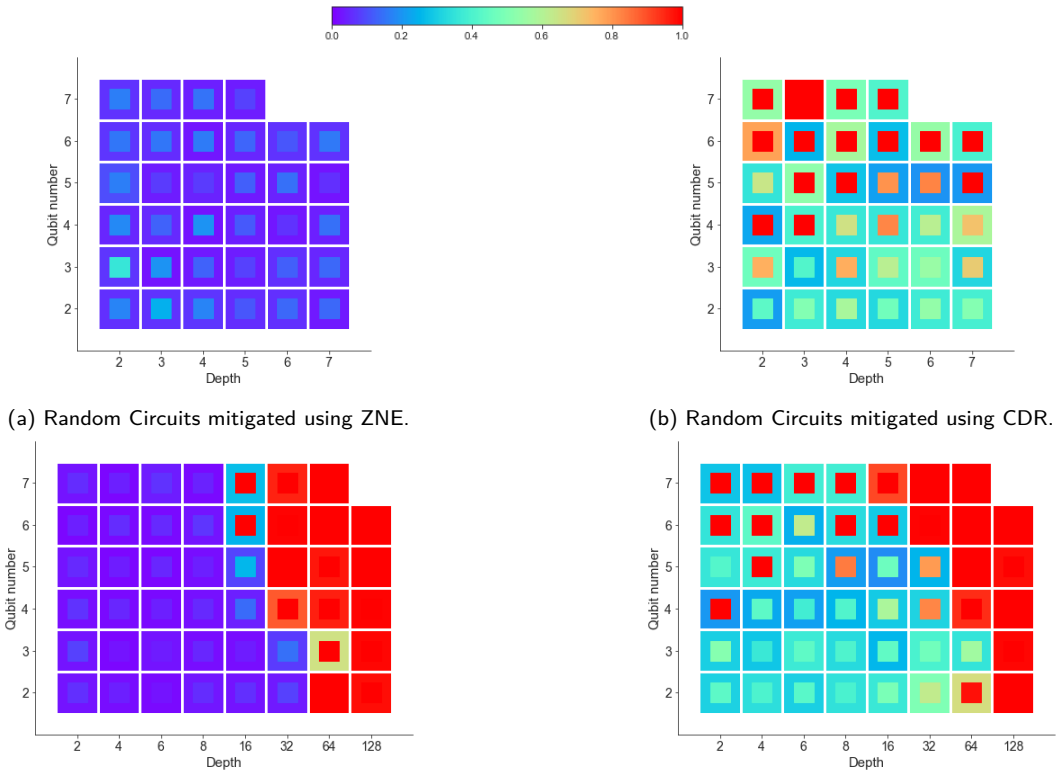
Therefore the only condition left to ensure that Lemma E.1 can be directly applied to $\epsilon(O)$ is that the coefficient of variation $\delta_Y = \frac{\sigma_N}{\mathbb{E}\langle O\rangle_N - \langle O\rangle}$ is sufficiently small. As in the main text if $\rho = U_1 \,...\, U_d\rho_0 U_d^\dagger \,...\, U_1^\dagger$ is the target quantum state $\langle O\rangle = Tr(O\rho)$ and its noisy implementation $\tilde{\rho} := \mathcal{E}_1 \circ \mathcal{U}_1 \circ ... \circ \mathcal{E}_d \circ \mathcal{U}_d(\rho_0)$ then $\mathbb{E}\langle O\rangle_N = \text{tr}(O\tilde{\rho})$. Now we are looking at the difference,

$$\mu_N = |\text{tr}(O(\rho - \tilde{\rho})| \leq ||O||_\infty ||\rho - \rho_0||_1 \tag{26}$$

$$\leq ||O||_\infty ||\mathcal{U}_1 \circ ... \circ \mathcal{U}_d - \mathcal{E}_1 \circ \mathcal{U}_1 ... \circ \mathcal{E}_d \circ \mathcal{U}_d||_\diamond \tag{27}$$

where we have used the Holder inequality in the first bound, and then taken a supremum over all possible ρ to bound by the diamond norm $||\Phi||_\diamond = sup_{X:||X||_1 \leq 1} ||(\Phi \otimes \mathbb{I})(X)||_1$. For conditions of Lemma E.1 to be met we want that $\mu_N \geq \frac{\sigma_N}{\lambda}$ for some $\lambda \leq 1$. This along with sub-multiplicatively of the diamond norm implies that

$$\sigma_N \leq \lambda ||O||_\infty de_{\max} \tag{28}$$



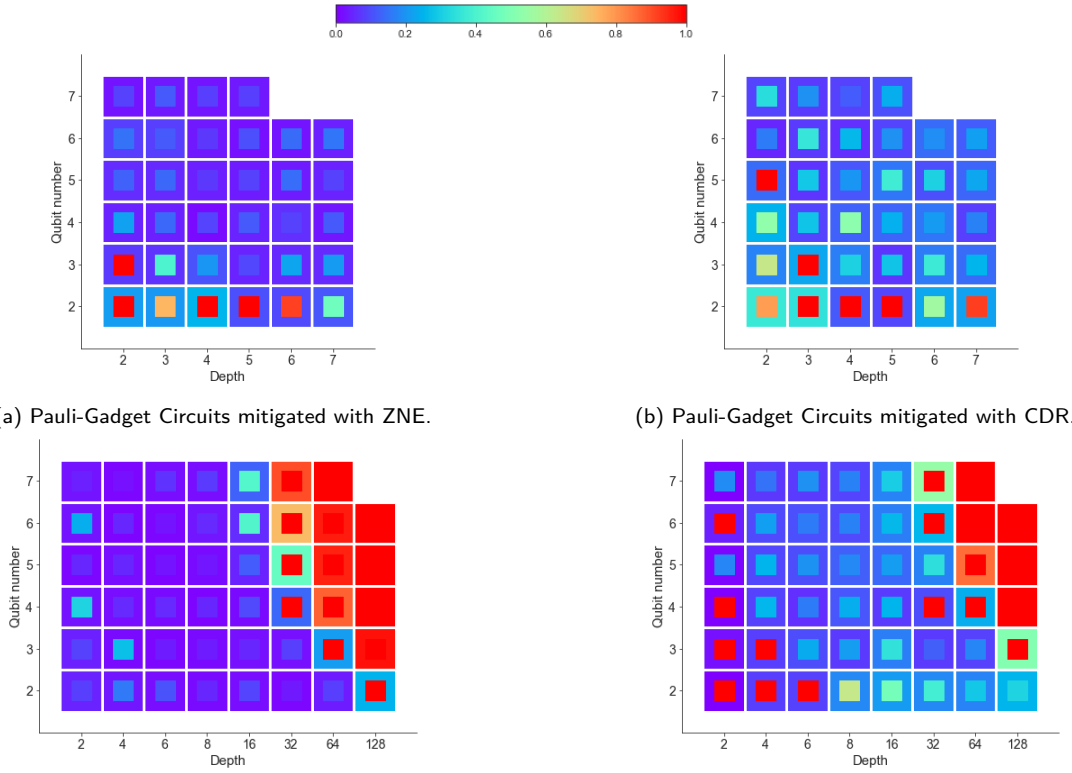
(a) Random Circuits mitigated using ZNE.

(b) Random Circuits mitigated using CDR.

(c) Mirrored Random Circuits mitigated using ZNE.

(d) Mirrored Random Circuits mitigated using CDR.

Figure 11: **Random Circuits run on a depolarising noise simulator**. Each square presents the results of 10 circuits. 500,000 shots are taken in total for each circuit by ZNE and CDR, and 100,000 in the case of the un-mitigated runs.



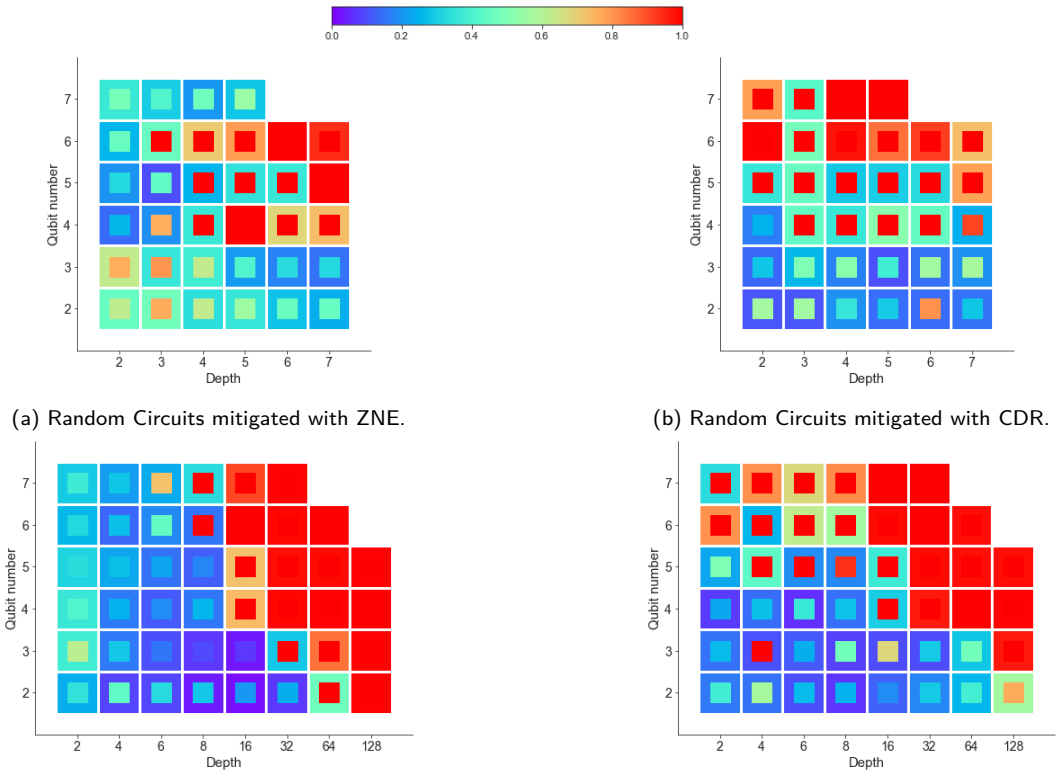
(a) Pauli-Gadget Circuits mitigated with ZNE.

(b) Pauli-Gadget Circuits mitigated with CDR.

(c) Mirrored Pauli-Gadget Circuits mitigated with ZNE.

(d) Mirrored Pauli-Gadget Circuits mitigated with CDR.

Figure 12: **Pauli-Gadget Circuits run on a depolarising noise simulator**. Each square presents the results of 10 circuits. 500,000 shots are taken in total for each circuit by ZNE and CDR, and 100,000 in the case of the un-mitigated runs.



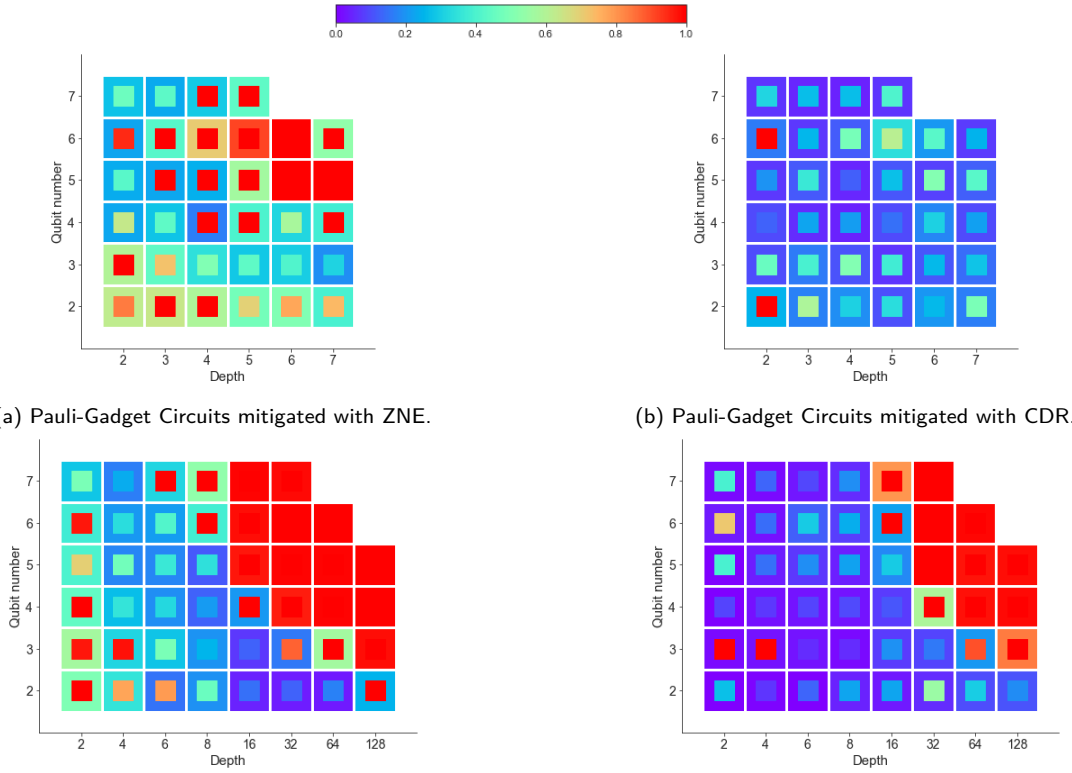
(a) Random Circuits mitigated with ZNE.

(b) Random Circuits mitigated with CDR.

(c) Mirrored Random Circuits mitigated with ZNE.

(d) Mirrored Random Circuits mitigated with CDR.

Figure 13: **Random Circuits on emulator backend.** Each square presents the results of 10 circuits. 160,000 shots are taken in total for each circuit by ZNE and CDR, and 32,000 in the case of the un-mitigated runs.



(a) Pauli-Gadget Circuits mitigated with ZNE.

(b) Pauli-Gadget Circuits mitigated with CDR.

(c) Pauli-Gadget Circuits mitigated with ZNE.

(d) Pauli-Gadget Circuits mitigated with CDR.

Figure 14: **Pauli-Gadget Circuits on emulator backend.** Each square presents the results of 10 circuits. 160,000 shots are taken in total for each circuit by ZNE and CDR, and 32,000 in the case of the un-mitigated runs.

where $e_{\max}$ is maximal (gate) error rate. Since σ_N^2 is the variance of the sample mean estimator for the noisy expectation value $\langle\hat{O}\rangle_N$, it will depend on the number shots n_0 so that $\sigma_N = \frac{\sigma_0}{\sqrt{n_0}}$, where σ_0 is the variance of an individual (i.i.d) sample of a single measurement. The condition Eq. (28) becomes

$$\frac{\sigma_0}{\sqrt{n_0}} \leq \lambda\, d\, e_{\max}\, ||O||_\infty multiplicatively \tag{29}$$

In particular this shows that, if one is to ensure that the relative error of mitigation follows a normal distribution under finite sampling then one must have the above trade-offs between the number of shots and depth. The condition also puts a lower bound on the depth of the circuit.

E.4 Effect of Finite Sampling on the Classical Optimisation to Determine Fit Parameters

In this section we consider the effect of finite size sampling on fitting the parameters of the functional model $\mathfrak{F}$. This type of analysis depends not only on the form of the functional $\mathfrak{F}$ itself but also on the classical optimization algorithm used to fit the function with noisy data $\mathbf{D}$.

For simplicity, we will assume that the data processing step involves a linear least square optimisation. Recall the following analytical expression for the fit parameter in a linear model $y_i = Ax_i + B$ for K data samples

$$\hat{A} = \frac{\sum_{i=1}^K (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^K (x_i - \bar{x})^2} \tag{30}$$

$$\hat{B} = \bar{y} - \hat{A}\bar{x} \tag{31}$$

where the estimators minimise the mean square error $\sum_{i=1}^K (y_i - Ax_i - B)^2$.

As it proves valuable in calculating σ_{EM}^2, the variance in $\langle\hat{O}\rangle_{EM}$, note the following derivation of the variance in the estimator $\hat{A}$

$$\mathrm{Var}\left[\hat{A}\right] = \mathrm{Var}\left[\frac{\sum_{i=1}^K (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^K (x_i - \bar{x})^2}\right] \tag{32}$$

$$= \frac{\sum_{i=1}^K (x_i - \bar{x})^2 \mathrm{Var}\left[(y_i - \bar{y})\right]}{\left(\sum_{i=1}^K (x_i - \bar{x})^2\right)^2} \tag{33}$$

$$= \frac{1}{\left(\sum_{i=1}^K (x_i - \bar{x})^2\right)} \frac{K+1}{K} \sigma^2 \tag{34}$$

where we have: assumed $\mathrm{Var}\left[y_i\right] = \sigma^2$ for all i, employed $\bar{y} = \frac{1}{K}\sum_{i=1}^K y_i$, and used that x_i and $\bar{x}$ are constants and not random variables. Note further

$$\mathrm{Var}\left[\hat{B}\right] = \mathrm{Var}\left[\bar{y} - \hat{A}\bar{x}\right] = \frac{\sigma^2}{K} + \bar{x}^2 \mathrm{Var}\left[\hat{A}\right]. \tag{35}$$

In the examples considered below, for CDR and ZNE the finite size sampling errors imply that the y_i's can be viewed as normally distributed random variables.

E.4.1 Clifford Data Regression

We consider a linear functional model as in Eq. (8) with least square optimisation as a classical method to determine the model parameters. Then we have $\hat{A} = \hat{F}_1$, $\hat{B} = \hat{F}_0$, $x_i = D_i^c$, and $y_i = D_i^q$ and the error mitigated estimator is given by

$$\langle\hat{O}\rangle_{EM} = \frac{\langle\hat{O}\rangle_N - \hat{F}_0}{\hat{F}_1}. \tag{36}$$

If $\hat{F}_0$ and $\hat{F}_1$ were a constant fixed values, F_0 and F_1, then

$$\sigma_{EM}^2 = \frac{\sigma_N^2}{F_1^2}. \tag{37}$$

However, both are unbiased estimators constructed out of (noisy) sample mean estimators, so incur a variance due to a finite number of shots. Treating $\hat{F}_0$, $\hat{F}_1$ and $\langle\hat{O}\rangle_N$ as normally-distributed independent random variables gives

$$\sigma_{\mathrm{EM}}^2 = \mathrm{Var}\left[\frac{\langle\hat{O}\rangle_N - \hat{F}_0}{\hat{F}_1}\right] \tag{38}$$

$$\approx \frac{\sigma_N^2 + \mathrm{Var}\left[\hat{F}_0\right]}{\hat{F}_1^2} + \frac{\left(\langle\hat{O}\rangle_N - \hat{F}_0\right)^2 \mathrm{Var}\left[\hat{F}_1\right]}{\hat{F}_1^4} \tag{39}$$

$$= \frac{(|D^c|+1)\,\sigma_N^2}{|D^c|\hat{F}_1^2} + \frac{\left(\langle\hat{O}\rangle_N - \hat{F}_0\right)^2 + \hat{F}_1^2 \bar{D^c}^2}{\hat{F}_1^4} \mathrm{Var}\left[\hat{F}_1\right] \tag{40}$$

$$= \frac{(|D^c|+1)\,\sigma_N^2}{|D^c|\hat{F}_1^2} \left[1 + \frac{\left(\langle\hat{O}\rangle_N - \hat{F}_0\right)^2 + \hat{F}_1^2 \bar{D^c}^2}{\hat{F}_1^2 \left(\sum_{i=1}^{|D^c|} \left(D_i^c - \bar{D^c}\right)^2\right)}\right] \tag{41}$$

where we employ respectively Lemma E.1, Eq. (35), and Eq. (32), and use $|D^c|$ to denote the number of Clifford training circuits. Note that an honest treatment of Lemma E.1 would require knowledge of exact values of $\langle O\rangle_N$, F_0, and F_1, rather than the estimators we use here. As we do not have access to these exact values, we conjecture that this approximation is accurate in the limit of large numbers of shots.

E.4.2 Zero Noise Extrapolation

We consider exponential extrapolation and the parameters in the functional model $D_{\lambda_i} = F_0 e^{F_1 \lambda_i}$ with $\langle\hat{O}\rangle_{EM} = \hat{F}_0$ determined via linear least square optimisation. Linearisation of the exponential function gives $\hat{A} = \hat{F}_1$, $\hat{B} = \log\left(\hat{F}_0\right)$, $x_i = \lambda_i$, and $y_i = \log\left(D_{\lambda_i}^q\right)$. If again we treat $\hat{F}_1$ as being the constant value F_1 then we recover from Eq. (35) that

$$\mathrm{Var}\left[\log\left(\langle\hat{O}\rangle_{EM}\right)\right] = \frac{1}{|\lambda|} \mathrm{Var}\left[\log\left(\langle\hat{O}\rangle_N\right)\right] \tag{42}$$

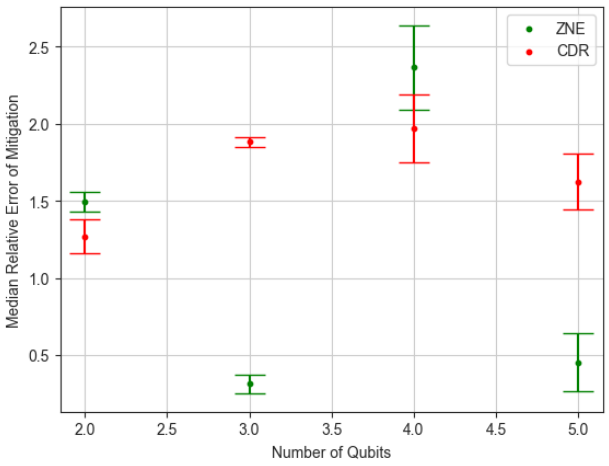


Figure 15: Data is as in Fig. 8a and Fig. 8b, with a qubit number of two. Error bars show one standard deviation.

where $|\lambda|$ is the number of noise scaling values, and where we have assumed

$$\mathrm{Var}\left[\log\left(D^q_{\lambda_i}\right)\right] = \mathrm{Var}\left[\log\left(\langle\hat{O}\rangle_N\right)\right], \ \forall i. \tag{43}$$

Assuming further that $\frac{\sigma^2_{EM}}{\langle\hat{O}\rangle^2_{EM}}, \frac{\sigma^2_N}{\langle\hat{O}\rangle^2_N} \ll 1$, gives

$$\frac{\sigma^2_{EM}}{\langle\hat{O}\rangle^2_{EM}} \approx \frac{1}{|\lambda|}\frac{\sigma^2_N}{\langle\hat{O}\rangle^2_N}. \tag{44}$$

However, it is more honest to consider $\hat{F}_1$ as a random variable, in which case

$$\frac{\sigma^2_{EM}}{\langle\hat{O}\rangle^2_{EM}} \approx \frac{1}{|\lambda|}\frac{\sigma^2_N}{\langle\hat{O}\rangle^2_N} + \bar{\lambda}\mathrm{Var}\left[\hat{F}_1\right] \tag{45}$$

$$= \frac{1}{|\lambda|}\frac{\sigma^2_N}{\langle\hat{O}\rangle^2_N} + \bar{\lambda}\frac{1}{\left(\sum_{i=1}^{|\lambda|}(\lambda_i - \bar{\lambda})^2\right)}\frac{\lambda+1}{|\lambda|}\sigma^2_N \tag{46}$$

E.4.3 Relative Error of Mitigation Variance

We combine the arguments in Appendices E.4.1 and E.4.2 with Eq. (13) to produce Fig. 15, which exemplifies well controlled variance in the case of our experiments. Here σ^2_N, the variance in the estimator of the noisy expectation value, is approximated by resampling the shots produced in the experiments of Figs. 8a and 8b. σ^2_{EM} is calculated using Eqs. (37) and (44).