Automated quantum error mitigation based on probabilistic error reduction

Benjamin McDonough
Department of Physics and Astronomy
Yale University
New Haven, Connecticut 06511, USA
ben.mcdonough@yale.edu

Andrea Mari
Unitary Fund
San Francisco, California 94104, USA
andrea@unitary.fund

Nathan Shammah
Unitary Fund
San Francisco, California 94104, USA
nathan@unitary.fund

Nathaniel T. Stemen
Unitary Fund
San Francisco, California 94104, USA
nate@unitary.fund

Misty Wahl
Unitary Fund
San Francisco, California 94104, USA
misty@unitary.fund

William J. Zeng
Unitary Fund
San Francisco, California 94104, USA
Goldman & Sachs
New York, NY, USA
will@unitary.fund

Peter P. Orth
Ames National Laboratory
Ames, Iowa 50011, USA
Department of Physics and Astronomy
Iowa State University
Ames, Iowa 50011, USA
porth@iastate.edu

***Abstract*—Current quantum computers suffer from a level of noise that prohibits extracting useful results directly from longer computations. The figure of merit in many near-term quantum algorithms is an expectation value measured at the end of the computation, which experiences a bias in the presence of hardware noise. A systematic way to remove such bias is probabilistic error cancellation (PEC). PEC requires a full characterization of the noise and introduces a sampling overhead that increases exponentially with circuit depth, prohibiting high-depth circuits at realistic noise levels. Probabilistic error reduction (PER) is a related quantum error mitigation method that systematically reduces the sampling overhead at the cost of reintroducing bias. In combination with zero-noise extrapolation, PER can yield expectation values with an accuracy comparable to PEC. Noise reduction through PER is broadly applicable to near-term algorithms, and the automated implementation of PER is thus desirable for facilitating its widespread use. To this end, we present an automated quantum error mitigation software framework that includes noise tomography and application of PER to user-specified circuits. We provide a multi-platform Python package that implements a recently developed Pauli noise tomography (PNT) technique for learning a sparse Pauli noise model and exploits a Pauli noise scaling method to carry out PER. We also provide software tools that leverage a previously developed toolchain, employing PyGSTi for gate set tomography and providing a functionality to use the software Mitiq for PER and zero-noise extrapolation to obtain error-mitigated expectation values on a user-defined circuit.**

***Index Terms*—quantum noise tomography, quantum error mitigation, noisy intermediate-scale quantum computing, probabilistic error reduction, zero noise extrapolation**

I. Introduction and overview

Hardware noise introduces unwanted bias into an expectation value measured on a quantum computer, restricting the applicability of many quantum algorithms on current quantum devices. Combating this problem has led to the development of probabilistic error cancellation (PEC) [1]–[3], a systematic method to remove the noise-induced bias. PEC requires a representation of the desired unitary channel as a linear combination of noisy channels that can be implemented on the hardware, which demands a precise characterization of the noise. Using the linearity of the expectation value, one can then express any ideal value without bias in terms of values obtained from instances of circuits with noisy gates. However, the number of noisy circuits required to represent an ideal circuit increases exponentially with the circuit depth, which would result in an exponentially large number of expectation values being measured. To overcome this issue, the linear combination can be converted into a quasi-probability distribution (QPD), from which sampling circuits yields an unbiased estimator of the value [1]–[3]. Due to the presence of

This work was primarily supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Superconducting Quantum Materials and Systems Center (SQMS) under the contract No. DE-AC02-07CH11359 (B.M., P.P.O.). A.M., N.S., N.T.S., M.W., W.J.Z. were supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing under Award Number de-sc0020266 and by IBM under Sponsored Research Agreement No. W1975810.

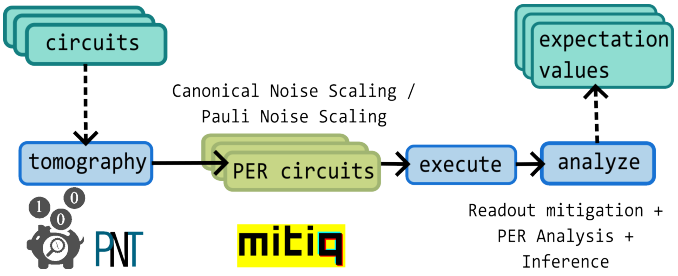

Fig. 1: Illustration of automated error mitigation protocol starting from user defined circuits and returning noise-mitigated expectation values. It includes a noise tomography step involving PNT or GST, whose results are used to generate sampled PER circuits via canonical or Pauli noise scaling. The circuits are executed with a user-specified interface to a quantum backend, and the resulting expectation values are adjusted according to the sampling overhead and the PER quasiprobability coefficient of the sampled term. Virtual ZNE is applied to approximate error-mitigated expectation values corresponding to noiseless input circuits.

negative coefficients in the expansion, the method experiences a sign problem, and requires an exponential sampling overhead to reduce the variance below a desired threshold. At current hardware noise levels, the sampling overhead limits PEC to circuits of modest depth [4]. To overcome this restriction, Mari *et al.* developed probabilistic error reduction (PER) [5], wherein the noise is only partially mitigated rather than fully canceled. By combining partial noise mitigation with virtual zero-noise extrapolation (vZNE), it was suggested that noiseless observables can be approximated at an accuracy similar to that of PEC, but with significantly reduced sampling costs (a related technique was also proposed in Ref. [6]).

Here, we describe and implement a framework for executing automated quantum error mitigation based on PER. We interconnect hardware noise characterization with the generation, sampling and analysis of PER mitigation circuits to produce expectation values with reduced bias, as illustrated in Fig. 1. We employ two separate methods of noise characterization: gate set tomography (GST) [7], [8] and a recently proposed cycle benchmarking technique for extracting a sparse Pauli noise model [9], [10], which we refer to as Pauli noise tomography (PNT). GST is implemented using the software package PyGSTi [11], which returns a set of noisy Pauli transfer matrices describing the set of implementable gates $\{\mathcal{O}_\alpha\}$. We convert these to superoperators and pass them into the open-source software package Mitiq [12], a software package containing a collection of quantum error mitigation techniques, to determine the decomposition of a set of desired noiseless unitary superoperators $\mathcal{G}_i$ as

$$\mathcal{G}_i = \sum_\alpha \eta_{\alpha,i} \mathcal{O}_\alpha \,. \tag{1}$$

Using the previously developed technique of canonical noise scaling [5], this decomposition is then used to create noise-scaled representations of the channel, $\mathcal{G}_i^{(\xi)}$, that depend on a tunable noise level parameter ξ. Circuits are sampled from these representations to obtain several noise-mitigated expectation values that are evaluated at different values of ξ. From these results we extrapolate the noiseless ($\xi \to 0$) expectation value using vZNE. Our contribution to this method is to develop software routines to facilitate this workflow of applying PER and vZNE using GST as implemented in PyGSTi for noise tomography to a user-specified circuit. We demonstrate this technique in an executable Python notebook, using Mitiq to produce a QPD representation with the noisy operators obtained from GST [13].

The second method we implement is a recently proposed scheme for efficient benchmarking of a Pauli-twirled noise channel and a technique for sampling from the noise inverse, which we extend to PER. [9], [10]. This method allows for robust characterization of hardware noise in large devices by keeping the number of measurements constant in the number of qubits. This is possible by placing weak assumptions on the level of correlation in the noise. This method compares favorably with full gate set tomography, which scales exponentially in the number of qubits. The PNT method has been used previously to perform PEC on IBM hardware [10]. Our key contributions are the following: 1) We combine the advantages of PNT with the sampling overhead reduction of PER and vZNE by extending this method to noise scaling, and 2) We provide a Python package implementing the combined PNT and Pauli noise scaling method as part of the general automated PER framework. The software automates all steps in the framework: Parsing an arbitrary circuit, generating tomography circuits, collecting and analyzing tomography data to obtain a noise model, generating noise-scaled circuit representations for PER, and running extrapolation to obtain noise-mitigated expectation values. The code is available in the accompanying repository [13].

In the remainder of the article, we first describe how to combine GST with canonical noise scaling and vZNE and show that this can reduce the sampling overhead compared to PEC. Then, we describe PNT, together with a demonstration of its usage in combination with PER and vZNE. We discuss in detail the developed software package for automated quantum error mitigation based on PNT and Pauli noise scaling to apply PER combined with vZNE for mitigation. Finally, we apply it to mitigate noise in a Trotter dynamics simulation of the transverse field Ising model.

Acronym	Meaning
PEC	Probabilistic error cancellation
PER	Probabilistic error reduction
GST	Gate set tomography
PNT	Pauli noise tomography
vZNE	virtual Zero-Noise Extrapolation
QPD	Quasi-probability distribution

TABLE I: List of acronyms used in the text.

II. Gate set tomography and and probabilistic error reduction

A. Gate set tomography

Gate set tomography (GST) [7], [8] is a method for characterizing noise associated with a set of gate operations capable of preparing a complete set of density matrices. The details of the long-sequence GST used here, including the choice of gate strings that maximally amplify the errors and a choice of suitable SPAM gates, are implemented by the software package PyGSTi [11]. PyGSTi provides model packs with pre-computed gate strings and SPAM gates. Here, we use the *sm1Q-XZ* model pack to reconstruct the single qubit gate set $\mathcal{G} = \{RX(\frac{\pi}{2}), RZ(\frac{\pi}{2})\}$ on Rigetti Aspen-11, a cloud-accessible superconducting-circuit quantum processing unit (QPU). The maximum depths for the long sequence gate strings were chosen to be $\{2, 4, 8, 16, 32\}$, and the GST experiment include a total number of 550 circuits, each of which was run at 1000 shots.

The output of PyGSTi is the set of operators represented as Pauli-transfer matrices (PTMs), which can be further processed as part of an error mitigation workflow using Mitiq. GST results exhibit a gauge degree of freedom related to the uncertainty in both state-preparation and measurement processes, which results in a set of PTMs which are similar to the ideal operations up to conjugation by an invertible matrix. The gauge freedom is by definition not detected in the measurement of expectation values, but it does affect the decomposition of ideal gates in terms of noisy ones in Eq. (1) and thus the sampling overhead of PEC and PER. Here, we use the knowledge that the RZ gate is virtualized on many platforms, including the Rigetti hardware, and use PyGSTi's inbuilt gauge optimization methods to choose a gauge such that the RZ PTM is noiseless. Full gate set tomography scales exponentially in complexity with the number of qubits, which in practice restricts the number of qubits operated on by a gate set to below three. An avenue for future work is to explore variants of GST protocols capable of characterizing noise in larger devices [8], [14]. This is typically made possible by placing stronger assumptions on the locality of the noise, which may limit the effectiveness of these protocols for PEC due to the untracked level of crosstalk in the device [15], [16].

B. Canonical noise scaling and vZNE

Noise characterization is only the first step for noise-sensitive quantum error mitigation. To proceed, we first convert the PTMs of the noisy gate operations obtained from GST into a superoperator representation. We then feed these into Mitiq to obtain a quasi-probability representation of ideal noiseless gates, as shown in Eq. (1). Applying canonical noise scaling [5] to this representation, it is then straightforward to derive a decomposition of noise-reduced gates $\mathcal{G}_i^{(\xi)}$ with noise strength $\xi \in [0, \frac{\gamma_i+1}{\gamma_i-1}]$:

$$\mathcal{G}^{(\xi)} = (\gamma^+ - \xi\gamma^-)\Phi^+ - (1-\xi)\gamma^-\Phi^- . \tag{2}$$

Here we have dropped the subscript i and separated the positive and negative coefficients η_α in Eq. (1) into $\gamma^+ = \sum_{\eta_\alpha>0}|\eta_\alpha| > 0$ and $\gamma^- = \sum_{\eta_\alpha<0}|\eta_\alpha|$, corresponding to the positive and negative volumes of the QPD. We defined $\gamma = \gamma^+ + \gamma^- = 1 + 2\gamma^-$, which is the sampling overhead at $\xi = 0$, corresponding to PEC. The overhead is determined by γ^-, which is referred to as the negativity of the QPD. We also introduced the completely-positive and trace preserving (CPTP) maps $\Phi^\pm = \sum_{\eta_\alpha \gtrless 0} \frac{|\eta_\alpha|}{\gamma^\pm}\mathcal{O}_\alpha$. The sampling overhead for $\xi \in [0, 1]$ is given by

$$\gamma^{(\xi)} = \gamma - \xi(\gamma - 1) . \tag{3}$$

This shows that PER provides a way to systematically reduce this overhead by removing the noise only partially. This reduction in overhead is shown explicitly in Fig. 2(a) for a noise-scaled reconstruction of an $RX(\frac{\pi}{2})$ gate with a noisy single-qubit gate set obtained using GST on Rigetti's Aspen-11, which yields $\gamma = 1.73$. The sampling overhead for a circuit of modest depth of $l = 8$ gates, $(\gamma^{(\xi)})^l$, is reduced from over 80 for PEC ($\xi = 0$) to about 5 for $\xi = 0.8$. We note that enforcing a noiseless RZ gate increases the overhead to $\gamma = 2.67$, leading to an even more steeply increasing sampling cost.

As shown in Fig. 2(b), although the price of reducing the overhead is additional bias, we can improve the estimate of the desired noiseless expectation values by leveraging the fact that expectation values converge to their ideal values at $\xi = 0$. Several values obtained through PER at different noise levels ξ can thus be used to extrapolate to the zero-noise limit. This vZNE procedure [5] can yield a zero noise estimator close to PEC at much lower sampling costs.

Finally, we note that Eq. (2) can also be used to scale up the noise when choosing $\xi \in [1, \frac{\gamma_i+1}{\gamma_i-1}]$, which does not incur any additional sampling overhead. Within ZNE one can thus combine estimator values at both reduced and increased noise levels, which can further improve the ZNE fit at low additional sampling costs (see Fig. 2).

III. Pauli noise tomography and probabilistic error reduction

We provide software that implements a recent noise characterization protocol based on cycle benchmarking [9], [10], which we refer to as Pauli noise tomography (PNT). This technique involves applying a Pauli twirl surrounding the Clifford entangling gates contained in layers of the circuit. This has been shown to convert an arbitrary noise channel into a Pauli channel [9], [10], [17]. The resulting Pauli fidelities characterize the twirled noise. PNT is able to achieve a constant scaling in the number of qubits, which makes this benchmarking method practically applicable to larger systems. This is achieved in part by modeling the twirled noise channel with a Lindbladian whose quantum jump operators are proportional to Pauli terms with support on qubits that are physically connected to each other in the quantum processing unit (QPU). This physically motivated assumption greatly reduces the complexity of the algorithm. After extracting

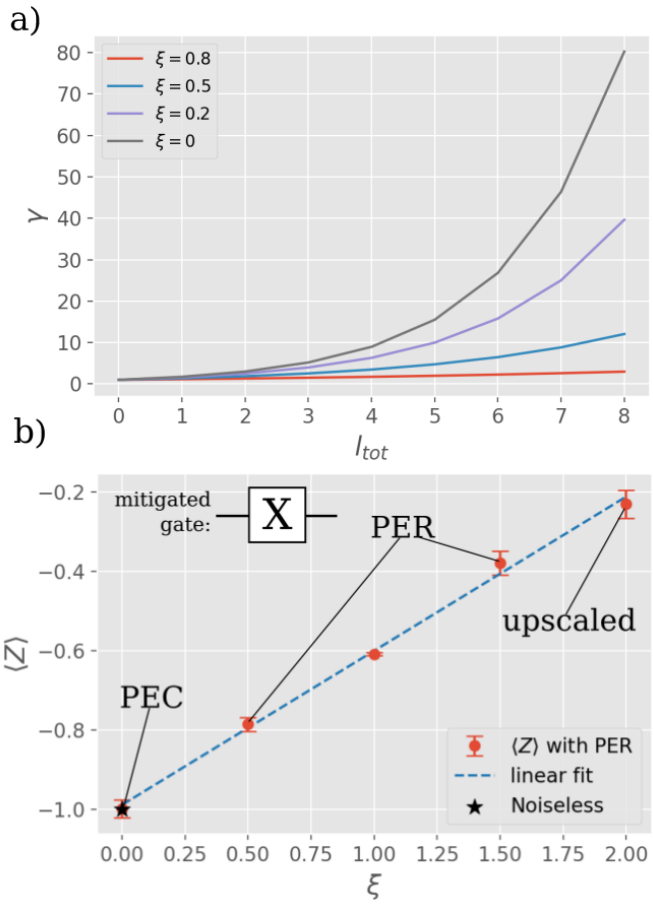

Fig. 2: (a) Sampling overhead $\gamma_{\text{tot}}^{(\xi)} = (\gamma^{(\xi)})^l$ versus circuit depth l for different noise strengths ξ using a canonical noise scaled representation derived from GST. Here, $\gamma_0 = 1.73$ as obtained from GST on Rigetti Aspen-11 and $\gamma^{(\xi)}$ is defined in Eq. (3). The figure demonstrates that a realistic sampling overhead γ of PEC on current hardware can be prohibitive for circuits of even modest depth l_{tot}, which validates the need for methods to reduce the overhead of noise-sensitive error mitigation such as PER and vZNE. (b) PER results for $\langle Z \rangle$ after application of a noisy X gate to $|0\rangle$. This was simulated for a random Pauli noise model chosen to have an overhead matching that obtained from GST on the Rigetti QPU. The decomposition of the noise reduced operator $\mathcal{G}^{(\xi)}$ in Eq. (2) can be obtained from GST using Mitiq's optimal representation algorithm and applying canonical noise scaling or from partially inverting the Pauli noise model Λ obtained from PNT (discussed in detail in Sec. III-A).

a sparse Pauli noise model, PEC circuits can be generated using an efficient procedure to sample from the inverse of the noise channel [10]. We here extend this approach to PER by constructing a *partially inverted* noise channel, enabling the level of noise to be controlled by a parameter ξ. We demonstrate the efficacy of this method in an accompanying tutorial notebook, where it is applied to a Trotter dynamics simulation of four qubits on a noisy backend emulator [13].

A. Pauli noise tomography

The advantage of PNT is its ability to capture correlated noise with low algorithmic complexity. The method works by decomposing circuits into "dressed" layers, which consist of a sequence of layers of single-qubit gates that can be compiled together, followed by a layer of self-adjoint Clifford gates with disjoint support. On platforms that support single-qubit rotations and the CNOT or CZ gate, any circuit can be decomposed into this form.

This reduction in complexity is achieved by first Pauli-twirling the noise channel Λ_l associated with a layer l. The twirl is implemented by sampling random Pauli operators around the noisy layer such that they have no logical effect on the circuit. This has the effect of diagonalizing Λ (dropping the subscript l) in the Pauli basis. This can be expressed symbolically as

$$\mathbb{E}_i[P_i\Lambda(P_i\rho P_i)P_i] = \sum_k c_k P_k \rho P_k \equiv \Lambda^{\mathbb{P}^n}(\rho)\,. \tag{4}$$

Once the channel is diagonalized, it can be characterized by the fidelities f_a, which are the diagonal elements of the Pauli transfer matrix:

$$f_a = \frac{1}{d}\,\mathrm{Tr}\Big(P_a\Lambda^{\mathbb{P}^n}(P_a)\Big)\,. \tag{5}$$

For a Pauli channel, these fidelities can be easily measured. If $|+\rangle_a$ is a $+1$ eigenstate of P_a, then we observe

$$\frac{1}{d}\,\mathrm{Tr}\Big(P_a\Lambda^{\mathbb{P}^n}(P_a)\Big) = \mathrm{Tr}\Big(P_a\Lambda^{\mathbb{P}^n}(|+\rangle\langle+|_a)\Big)\,. \tag{6}$$

Since the Pauli matrices P_a are eigenvectors of the twirled noise channel (dropping the $\mathbb{P}^n$ superscript from here on), the fidelities can be determined through exponential fits. However, since the noise is attached to a self-adjoint layer of Clifford gates $\mathcal{C}$, applying an even number of these layers results in the measurement of fidelity pairs:

$$\frac{1}{d}\,\mathrm{Tr}\big(P_a(\Lambda\circ\mathcal{C})^{2n}(\rho)\big) = (f_a f_a')^n\,. \tag{7}$$

where f_a' is the fidelity of $P_a^C = \mathcal{C}(P_a) = CP_aC^\dagger$. In the most general case, there are still exponentially many fidelities f_a requiring measurement. Under the assumption that noise correlations are strongest when there exist physical connections between qubits, the twirled noise is modeled using a master equation where the quantum jump operators $\sqrt{\lambda_k}P_k$ are chosen to be only those Paulis which have support on neighboring qubits [10]. It can be shown that the noise channel, written as a superoperator, assumes the following form:

$$\Lambda = \prod_k (w_k\mathcal{I} + (1-w_k)\mathcal{P}_k)\,, \tag{8}$$

where $w_k = \frac{1}{2}(1+e^{-2\lambda_k})$, $\mathcal{I}$ is the identity, and $\mathcal{P}_k\rho = P_k\rho P_k^\dagger$. Given this form of the noise, the fidelitites f_a can be related to the coefficients λ_k by the equation

$$f_a = \prod_{\{P_a,P_k\}=0}(1-2w_k) = \exp\left(-2\sum_{\{P_a,P_k\}=0}\lambda_k\right)\,. \tag{9}$$

Only the fidelities f_a corresponding to the terms in the sparse model require measurement to reconstruct the model. Many QPU architectures exhibit a local qubit connectivity, often restricted to nearest-neighbor pairs for which the number of terms in the model scales linearly in the number of qubits. The ability to simultaneously measure commuting terms results in the additional improvement to constant scaling. Since PNT characterizes the noise that is associated with a layer of Clifford gates, methods such as Trotter time evolution or variational quantum circuits that involve repetition of identical circuit layers also exhibit a constant scaling in the depth of the circuit.

In Fig. 3 we show PNT results for a randomly generated Pauli noise model. Since Pauli twirling is shown to convert arbitrary noise into Pauli noise, this efficacy extends to more realistic noise models as well. We observe excellent agreement between the simulated fidelities and the fidelities obtained via PNT for a layer consisting of a CNOT gate. Once the fidelities f_a are obtained, a matrix M can be constructed via $[M]_{ab} = \langle P_a, P_b \rangle_{sp}$, where $\langle \cdot, \cdot \rangle_{sp}$ refers to the symplectic inner product, which is zero if $[P_a, P_b] = 0$ and one otherwise. The vector of fidelities $\boldsymbol{f}$ is then related to the vector of noise model parameters $\boldsymbol{\lambda}$ via

$$2M\boldsymbol{\lambda} + \ln(\boldsymbol{f}) = 0\,. \tag{10}$$

The λ_k can be approximated from the measurements of f_a using a non-negative least squares algorithm, yielding a sparse Pauli noise model parametrized by λ_k (or equivalently w_k).

B. Partial Pauli noise inversion

We incorporate PNT into the PER error mitigation framework by generalizing the noise inversion through sampling from a QPD representation as discussed in [10] to performing a partial inverse. A partial inverse of the noise channel can be constructed with the form

$$\Lambda^{(\xi)} = \gamma^{(\xi)} \prod_k \left(w_k^{(\xi)} \mathcal{I} + \mathrm{sgn}(\xi - 1)(1 - w_k^{(\xi)}) \mathcal{P}_k \right). \tag{11}$$

Here, $w^{(\xi)} \equiv \frac{1}{2}(1 + e^{-2|1-\xi|\lambda_k})$ and the sampling overhead

$$\gamma^{(\xi)} = \begin{cases} \exp[2(1-\xi)\sum_k \lambda_k] & \xi < 1 \\ 1 & \xi \geq 1 \end{cases} \tag{12}$$

quantifies the variance in the estimator. This channel $\Lambda^{(\xi)}$ exhibits similar useful properties as we observed within canonical noise scaling in Sec. II-B. For example, upscaling the noise does not incur any sampling overhead yet can still provide useful information for vZNE. For intermediate values of the noise $0 < \xi < 1$, the overhead interpolates exponentially between unity and the PEC value, which enables a substantial reduction in the number of circuits that need to be executed. When $\xi \to 0$, we have $\Lambda^{(\xi)} \to \Lambda^{-1}$, corresponding to PEC. On the other hand, when $\xi \to \infty$, $\Lambda^{(\xi)}$ approaches maximal depolarizing noise, which maps every density matrix to the identity. Lastly, the product form of this partial inverse ensures that this method retains the efficient sampling from the inverse

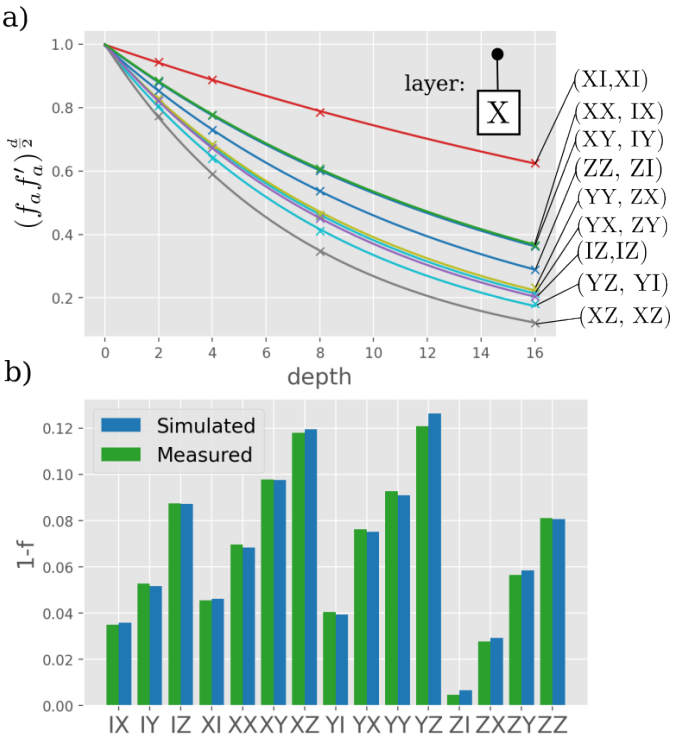

Fig. 3: PNT results for a layer consisting of a CNOT gate under a randomly generated Pauli noise model composed with an amplitude damping noise channel with $p = 0.01$. We use 32 samples for the Pauli twirl and run different circuits depths for 250 shots each. Panel (a) shows that the fidelity pairs $f_a f'_a$ of all Pauli operators on two qubits decay exponentially with circuit depth. The amplitude damping noise was included to show that the twirling properly diagonalizes the channel as evidenced by the exponential decays observed in the figure. Panel (b) compares the measured fidelity pairs to the products of diagonal elements of the twirled channel transfer matrix. The agreement between the measured and ideal values shows the efficacy of PNT.

discussed in [10]. The sampling procedure is described in Algorithm 1.

The total overhead is a product of the overhead of the individual layers l, $\gamma_{\mathrm{tot}}^{(\xi)} = \prod_l \gamma_l^{(\xi)}$, and so it still scales exponentially in the number of layers. However, by making this exponential scaling weaker, one can extend the practical application of error mitigation to larger circuits. Implementation details are discussed in the following section.

IV. Software for Automated Error Mitigation

To make the techniques described above practical for the end user, we here present a software package to automate the implementation of PER, from tomography to extrapolation. Our first attempt at such an automated framework is an object-oriented interface written in Python automating the PNT and Pauli noise scaling process. The chief goal of this

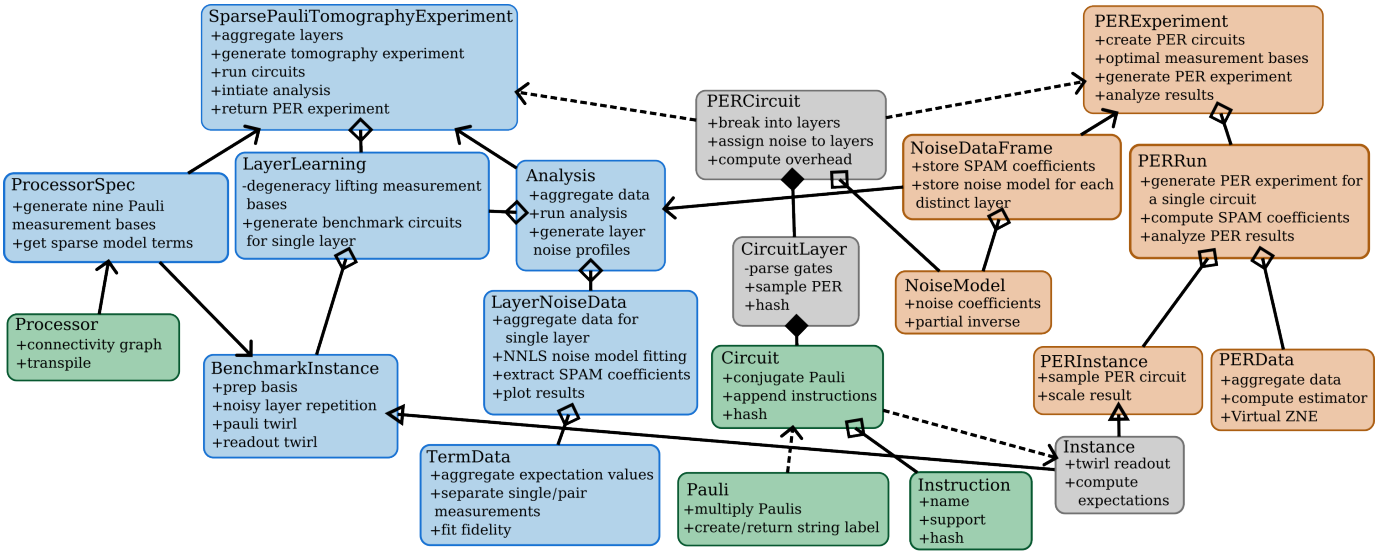

Fig. 4: UML diagram describing software tools for automated error mitigation. The solid lines with arrowheads represent association, the dotted lines with solid arrowheads represent dependency, the solid lines with triangular heads represent implementation, the lines with closed diamond heads represent composition, and the open diamond heads represent aggregation. Objects involved in tomography are shown in blue, objects in orange are involved in PER, and gray objects are used by both. The green objects are wrappers for external implementations (such as Qiskit) and require implementations with external dependencies.

Algorithm 1 Description of PER routine

Input: Circuit with layers $l \in \{1, .., l_{tot}\}$, each with noise model parameters $\{w_{l1}^{(\xi)}, ..., w_{ln}^{(\xi)}\}$

Output: A sample of the PER expectation value (before readout error mitigation)

Let $\alpha \equiv 1$
for $l \in \{1, \ldots, l_{\text{tot}}\}$ **do**
    Compose layer l into circuit
    for $k \in \{1, \ldots, n\}$ **do**
        Sample I with probability $w_{lk}^{(\xi)}$ and P_k otherwise
        Multiply α by $\gamma_l^{(\xi)}$
        if P_k was sampled **then**
            Multiply α by -1
        end if
        Compose sampled operator into circuit
    end for
end for
Run the circuit and get the expectation value
if $\xi < 1$ **then**
    Scale result by α
end if

implementation is to allow the user to apply this technique in its entirety without being burdened by the details. This section describes the functionality of this package to automate the process of performing tomography and using it to carry out PER, ultimately obtaining an error-mitigated set of desired expectation values.

At the top level, the software is divided into two parts: (i) tomography and (ii) PER. The program is intended to be easily extended to different platforms such as PyQuil or Circ, interacting with the native implementation through the abstract classes `Circuit`, `Processor`, `Pauli`, and `Instruction`. A Unified Modeling Language (UML) diagram highlights these objects (green) in Fig. 4. These classes are wrappers for objects and behaviors common to many quantum development toolkits, and can be overridden to provide support for another API's. This allows circuits to be run in their native representation without any conversion. Currently only the Qiskit interface has been implemented.

A. *Pauli Noise Tomography software tools*

The method used for PNT is described in Ref. [10]. To begin the process, the `SparsePauliTomographyExperiment` class is initialized with a list of circuits, a mapping of algorithm qubits to physical qubits, and a quantum backend. The experiment class initializes an instance of `ProcessorSpec`, which uses the coupling map from the backend to generate a list of the Pauli terms with support on neighboring qubits. Using the sweeping algorithm described in Ref. [10], this object chooses the nine optimal measurement bases from which simultaneous measurements can be used to obtain fidelities of all Pauli terms in the sparse model. Then, the `PERCircuit` class separates each circuit into layers that have the form of any number of single-qubit gates followed by a layer of self-adjoint Clifford gates with disjoint supports. The noise of one of these layers is assumed to be determined just by the Clifford gates in the layer. The self-adjoint Clifford layers, from here

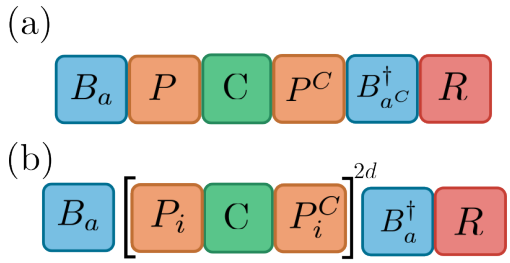

Fig. 5: Illustration of the two forms of benchmark circuits. The B_a gates change from the computational basis into the Pauli basis P_a being benchmarked. The Pauli twirl operator P is sampled at random from the Pauli group. C is the Clifford layer, and the superscript $(\cdot)^C$ represents conjugation by the Clifford layer. The R gate is readout twirling, sampled at random from $\{I, X\}^{\otimes n}$ [18], [19]. Adjacent single-qubit gates are compiled together in a way that preserves the structure of the dressed layers. Panel (a) shows single-depth measurements that lift the degeneracy in the model. Panel (b) shows a noisy layer repeated an even number of times, resulting in benchmarking fidelity pairs.

on just "Clifford layers," are therefore hashed into a set to be benchmarked individually.

Next, the user can call the `generate` method on the experiment class. The generation procedure creates a `LayerLearning` object for each distinct Clifford layer, which are responsible for generating the benchmark circuits for tomography. Each learning procedure consists of two types of circuits. The first represents the measurement of fidelity pairs. These are the measurements that can be made at multiple depths and fit to exponential decays. When the fidelities of two model terms appear in a pair, a degeneracy is created in the model. This degeneracy is lifted by the second type of circuit, which consists of a single repetition of the noisy layer. The use of a single repetition causes the preparation and measurement bases to differ, which makes this type of measurement less resistant to state preparation errors. In addition to the nine Pauli bases chosen for the pair measurements, there are at most six bases that are needed to make the degeneracy-lifting measurements, independent of the number of qubits. The initialized `LayerLearning` objects choose these basis in relation to the corresponding Clifford layer.

Each `LayerLearning` generates a list of `BenchmarkInstance` objects representing the desired measurements, and each `BenchmarkInstance` generates the circuit corresponding to this instance. This includes readout twirling, following the method described in [19], where gates are randomly sampled from the set $\{I, X\}$ before measurement on each of the qubits, and then the twirl is inverted in software. This has the effect of diagonalizing the readout error in the computational basis. The basis preparation, measurement, and readout twirling gates are stored as metadata. The next step for the user is to call the `run` method on the experiment. This method accepts as input a user-defined "executor" function, which executes a list of circuits on the QPU and returns the results as a dictionary.

To complete the tomography procedure, the `analyze` method can be called. This initializes the `Analysis` class, which is constructed with the `LayerLearning` classes from the experiment containing the experiment parameters and the benchmark data. The `Analysis` class creates a dictionary of `LayerNoiseData` objects for processing the data associated with each distinct Clifford layer. Each of these has a list of `TermData` objects, which will store the expected data for each term in the sparse model. The `get_expectation(pauli)` method is called on each `BenchmarkInstance` and used to update the estimator in the `PauliTerm` objects for each of the measurements that can be made simultaneously. The `get_expectation(pauli)` method is responsible for untwirling the result of the readout using the stored metadata and returning the overlap with the desired Pauli term in the computational basis. Finally, the resulting values are sorted in each `TermData` object into the type and depth of the measurement, where "type" indicates whether the measurement is a fidelity pair or a single-depth measurement.

Once the `TermData` objects for each `LayerNoiseData` have been populated, the results can be used to fit the fidelities. The pair fidelities are determined first through fits to ae^{-b}, where a quantifies the combined SPAM errors and $e^{-b} = \sqrt{f_a f'_a}$ is the square root of the fidelity pair. The pair measurements are used to determine SPAM coefficients for each measurement basis in order to mitigate SPAM errors in the single-depth measurements. The degeneracy-lifting fidelities are determined directly from the estimators of the expectation values, and the SPAM coefficients from the fits are used to reduce SPAM errors. Since the pair measurements are assumed to be more accurate, the value of a single-depth measurement f_a is limited by the pair fidelity f'_a via the constraint $f_a f'_a \leq 1$ and an exception is logged if f_a violates this bound.

After this set of single and pair fidelities have been determined, each layer can be fit to the sparse noise model to determine the layer coefficients. This is done as described in [10]: one forms a vector $\boldsymbol{b}$ from the fit results, and forms lists F_1 and F_2 containing Pauli operators. The entries of $\boldsymbol{b}$ are either single fidelities $b_a = f_a$ or fidelity pairs $b_a = \sqrt{f_a f'_a}$. If b_a is a pair, then P_a is added to F_1 and P'_a is added to F_2. If b_a is not a pair, then P_a is added to both F_1 and F_2. From here, one constructs matrices M_1 and M_2 using the definitions $[M_1]_{ab} = \langle F_{1a}, F_{1b} \rangle_{sp}$ and $[M_2]_{ab} = \langle F_{2a}, F_{1b} \rangle_{sp}$.

The Pauli noise model coefficients $\boldsymbol{\lambda}$ are obtained from a numerical solution of [cf. Eq. (10)]

$$(M_1 + M_2)\boldsymbol{f} + \ln(\boldsymbol{\lambda}) = 0\,, \tag{13}$$

where the logarithm is taken elementwise. The result of the fit is a `NoiseModel` object, and all of these are composed in a `NoiseDataFrame` object, which stores them as a dictionary with Clifford layers as keys. In addition, the SPAM coefficients for each single-weight Pauli measurement are

averaged together and stored to model the readout error for use in mitigation. This `NoiseDataFrame` object forms the link between the tomography and PER portions of the protocol.

The software package also provides several visualization tools to plot results of different steps of the process. Within the tomography procedure, there are three different plots implemented. The first is the plot of the exponential decays of the Pauli fidelity pairs with increasing circuit depth. This can be used to see if enough samples were taking from the twirl to properly diagonalize the channel. Next, the infidelities of Pauli operators can be plotted by specifying a list of qubits as single-element tuples or qubit pairs as tuples. Lastly, the coefficients λ_k appearing in the generator can be plotted against each other. This can be especially helpful to compare the errors experienced by different qubits and identify the dominant sources of error.

B. Probabilistic error reduction software tools

The `NoiseDataFrame` object resulting from the tomography contains all the data needed to carry out PER. Calling the `create_per_experiment(circuits)` method on the experiment class passes this object to a new PER experiment with a set of desired circuits to mitigate. Upon initialization, the `PERExperiment` class passes each circuit to the `PERCircuit` class, which breaks each circuit into dressed layers of the form described in Sec. III-A.

The generation of the PER circuits is initiated by calling `generate` on the `PERExperiment` instance. The arguments of this method are the desired expectation values, the number of samples to take from the combined distribution of the partial noise inverse, the Pauli twirl, and the readout twirl, and finally, the different noise strengths at which to run the circuits. Then, the minimal set of measurements that can simultaneously reconstruct the desired expectation values is determined. A new `PERRun` class is instantiated for each of the circuits on which to run PER. Each `PERRun` object is responsible for creating the desired `PERInstances` representing the collection of a simultaneous subset of the desired expectation values at a particular noise strength. The sampling procedure is described in Sec. III-B.

Once the circuits have been generated, they can be executed using the same `executor` method used in the tomography section. The result of running each circuit is paired with the `PERInstance` that produced it for later analysis. Once the run is complete, the `analyze` method can be invoked on the PER experiment to process the data. This calls the `analyze` method on each of the `PERRuns` in the experiment, which in turn assigns the populated `PERInstance` objects to a `PERData` object for each expectation value that can be simultaneously determined from the instance.

Object `PERData` calls `get_adjusted_expectation` on each of the `PERInstance` objects. This method is responsible for converting the resulting expectation value into a PER estimator by rescaling the raw expectation value of the circuit with the sign recording the parity of the number of nonidentity operators sampled in Eq. (11) and the overhead

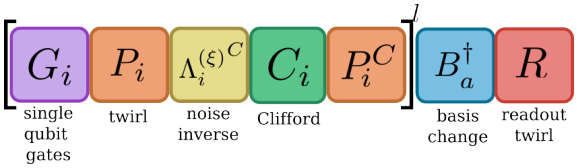

Fig. 6: Illustration of PER circuits. The user-specified circuit is parsed to obtain the single-qubit gates G_i and the Clifford layer C_i for each layer i in the circuit. Each layer is assigned a noise model produced by the tomography. B_a represents the gates used to change from the computational basis to the eigenbasis of P_a. The inverse is sampled at the desired noise strength along with the Pauli and readout twirling. The symbol $P^C = CPC^\dagger$ denotes the conjugation of P by the Clifford layer C.

$\gamma_l^{(\xi)}$ corresponding to the circuit layer to which it belongs. At this point, the SPAM coefficients obtained from tomography are used to perform readout error mitigation on the expectation value. Readout error mitigation is currently implemented under the assumption of uncorrelated readout noise to cut down on the number of circuits that need to be run, but as work such as [19] suggest, this assumption may be too strong, and this functionality should be improved in the future.

Finally, each `PerData` object performs vZNE on the expectation values taken at different noise strengths to yield a final PER estimate of the desired expectation values on the list of input circuits. The ansatz function used for the fit is ae^{-b}, where a is the ideal expectation value. There is reason to believe that this ansatz is at least approximately accurate for any circuit, but this merits future exploration.

The principal plotting tool in the PER module is the ability to show the convergence of the expectation value for different strengths of noise against the exponential fit. The `analyze` method returns the `PERRun` objects corresponding to different circuits, and the `get_result(pauli)` method can be used to obtain the `PERData` for a specific expectation value on the desired circuit. This object contains the data and plotting tools for this run.

V. Application Tutorial

We choose as our practical application a Trotter simulation of the postquench dynamics in the one-dimensional transverse-field Ising model (TFIM) with Hamiltonian

$$H = -J \sum_j Z_j Z_{j+1} - h \sum_j X_j \,. \tag{14}$$

This example is used for comparison to Ref. [10]. A single Trotter step can be constructed as a `QuantumCircuit` object in the form of Fig. 7. To simulate the evolution at different points in time, a method can be created to repeat this Trotter step n times by defining a function `trotterCircuit(n)`. A list of circuits corresponding to increasing Trotter steps can be generated:

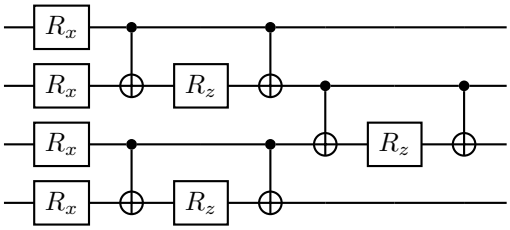

Fig. 7: The realization of a single Trotter step as a quantum circuit. Here we have defined $R_x \equiv RX(-2h\delta t)$ and $R_z \equiv RZ(2J\delta t)$

```
circuits = [
    trotterCircuit(n) for n in range(0, d)
]
```

Next, a backend should be initialized and used to transpile the circuits. In the tutorial notebook, we use `FakeVigoV2`. PNT is initialized via

```
from tomography.experiment \
import SparsePauliTomographyExperiment \
as pnt
experiment = pnt(
    circuits=circuits,
    inst_map=[0,1,2,3],
    backend=backend,
)
```

The transpiled circuits are passed to the experiment, along with the backend and a map from the virtual qubits in the circuit to the physical qubits on the backend. Next, the circuits for tomography can be generated with

```
experiment.generate(
    samples=32,
    single_samples=200,
    depths=[2,4,8,16],
)
```

These are the default options for the parameters. Increasing the samples will increase the goodness of the fit (for details see Sec. IV). The execution of circuits is exposed to the user by the use of a method which takes a list of quantum circuits and returns the result of executing these circuits on a quantum computer as a counts dictionary. An example of such an executor is

```
def executor(circuits):
    job = backend.run(circuits)
    return job.result().get_counts()
```

This executor can be used to call the `run` method of the experiment: `experiment.run(executor)` Once the execution is finished, the data can be analyzed by calling

```
experiment.analyze()
```

The experiment is now populated with all of the noise data needed to carry out PER. With this, the PER experiment can be set up for the desired circuits by calling

```
perexp = \
experiment.create_per_experiment(circuits)
```

The value we want to mitigate is the z-component of the magnetization, $M_z = \frac{1}{N}\sum_{i=0}^{N-1}\langle Z_i\rangle$. We can collect these expectation values by passing them to the `generate` method:

```
expectations = ["ZIII", "IZII",
                "IIZI","IIIZ"]
perexp.generate(
    expectations,
    samples=1000,
    noise_strengths=[0.5, 1, 2],
)
```

The bases to make as many of these measurements simultaneously as possible are chosen. For this example, this is simply the computational basis. Currently, for $d = 10$ Trotter steps at 1000 samples for three different noise depths, the generation of these circuits takes around an hour on a regular laptop computer. The time taken to generate these circuits is currently the biggest bottleneck to the execution time of the protocol. One potential solution would be to take advantage of the consistent form of the PER circuits by using parametric compilation to implement the twirl and partial noise inverse. This will be explored in future versions.

After generating the circuits, they can be run with

```
perexp.run(executor)
```

Finally, the results can be obtained by calling

```
circuit_results = perexp.analyze()
```

Each element in `circuit_results` now stores the results of PER containing each expectation value at each noise strength corresponding to the original `circuits` array that was passed in. Lastly, the magnetization can be computed for the i^{th} Trotter step by adding up the vZNE result of each expectation value:

```
res = circuit_results[i]
M_z[i] = sum([
    res.get_result(op).expectation
    for op in expectations
]) / N
```

The resulting array M_z is plotted versus time in Fig. 8(a), along with the noiseless value for comparison. Fig. 8(b,c) compares the distribution of PEC and PER estimators, highlighting the increase of the negativity of the QPD for smaller ξ and the resulting larger variance and sampling cost. This is further illustrated in Fig. 9, which shows vZNE for the individual $\langle Z_i\rangle$ at the final Trotter step of the simulation. At this depth $n = 15$, the number of PEC samples required to achieve a precision equal to δ is $\frac{\gamma^{(0)^2}}{\delta^2} \approx \frac{53}{\delta^2}$. In contrast, PER with the same precision at noise strengths $\xi \in [0.5, 1, 2]$ requires in total only $\frac{\gamma^{(0.5)^2}}{\delta^2} + \frac{2}{\delta^2} \approx \frac{9}{\delta^2}$ samples. While there may be some extra bias error introduced through vZNE, our results demonstrate that PER combined with vZNE can offer a significant advantage when the overhead is large.

VI. Conclusion and Outlook

Mitigating errors that occur in a noisy quantum computation is a key challenge that must be addressed in order to achieve

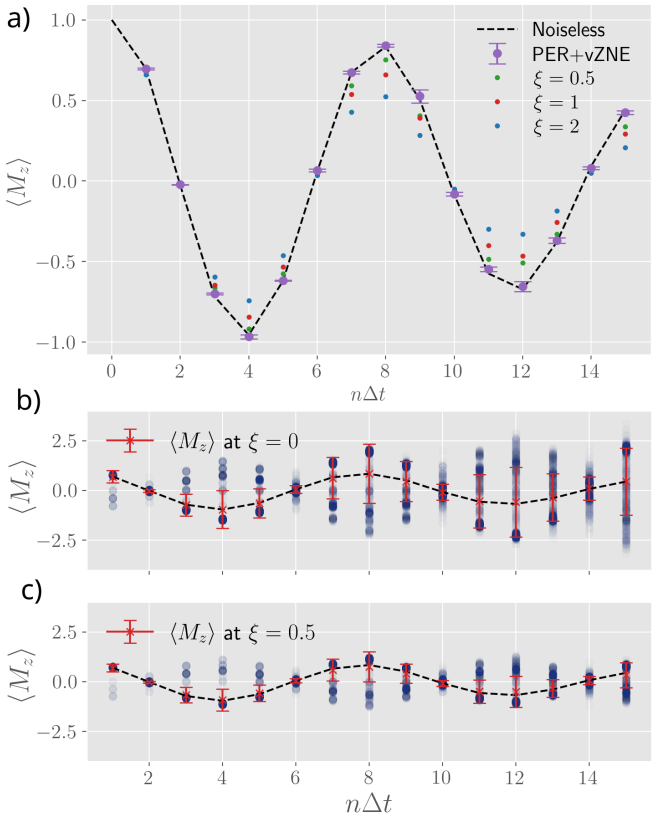

Fig. 8: PER results of total magnetization $\langle M_z(t) \rangle$ of the TFIM, prepared in the $|0\rangle$ initial state and evolved with under Hamiltonian with parameters $J = 0.15$, $h = 1$. The Trotter dynamics are simulated on the IBM noisy simulator `FakeVigoV2` using a Trotter stepsize $\Delta t = 0.2$. We simulate 1000 PER circuits, each is evaluated with 1024 shots. Panel (a) shows that vZNE with the noise levels $\xi \in [0.5, 1, 2]$ yields excellent agreement with the noiseless Trotter result. Readout error mitigation is used at all noise levels. Panel (b) shows the individual estimators (blue) and their average with standard deviation (red crosses) at $\xi = 0$ (upper plot) and $\xi = 0.5$ (lower plot). The variance increases as the noise strength approaches zero, and the estimator values approach the noiseless value. The overhead at $\xi = 0$ after 15 Trotter steps is $\gamma^{(0)} = 7.25$, while it is only $\gamma^{(0.5)} = 2.69$ at $\xi = 0.5$.

quantum advantage when full quantum error correction is not available. The use of quantum error mitigation is also expected to extend into the era of fault-tolerant quantum computing with logical qubits as it can lower the overhead for quantum error correction [20]–[22]. While hardware improvements that reduce gate error rates are crucial, the development of controlled and automated error mitigation software provides progress towards performing more complex quantum calculations. Since the controlled mitigation of quantum errors necessarily includes noise characterization as the first step, we advocate for the development of error mitigation software tools that combine noise characterization with mitigation capabilities.

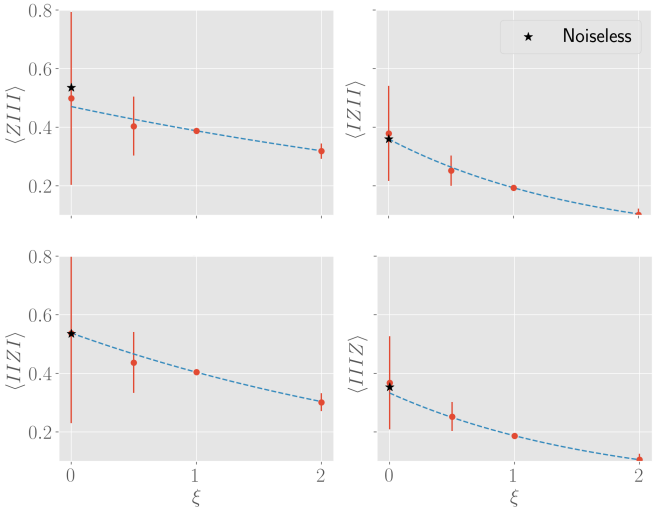

Fig. 9: Pauli Z expectation values of individual qubits at $n = 15$ Trotter steps as a function of noise strength ξ. We use 1000 PER circuits, each evaluated with 1024 shots. Variance increases and bias decreases for smaller ξ. vZNE with exponential fit using $\xi \in \{0.5, 1, 2\}$ yields an expectation value of similar accuracy as PEC. The star denotes the noiseless result.

We here provide a Python notebook with a practical example of how to efficiently connect GST with PER using canonical noise scaling and vZNE by connecting two existing software toolkits, PyGSTi and Mitiq. We also develop a software framework that combines PNT with PER using partial noise inversion and vZNE for a user-defined circuit. Since the complexity of this method is constant with respect to the number of qubits and the circuit depth for circuits with many repeated layers, this approach can be applied for larger systems. We provide a tutorial notebook showcasing the use of this method in a Trotter simulation of four qubits over 15 Trotter steps. Our results demonstrate that PER combined with vZNE can yield results of the same accuracy as PEC, but at much smaller sampling costs. Finally, as increasingly complex error mitigation schemes are developed, multi-platform software tools provide key advantages for software users engaging on multiple quantum computing platforms. We hope that our work lays a foundation for integrating the proposed workflow of GST and canonical noise scaling into Mitiq, and for the development of other software suites connecting noise characterization and error reduction to automate the application of noise-sensitive error mitigation to near-term algorithms on noisy intermediate-scale quantum (NISQ) devices.

VII. Reproducibility

We have made our raw data, source code, and tutorial notebooks available via an online appendix [13] for the research community to reproduce or use.

References

[1] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Phys. Rev. Lett.*, vol. 119, no. 18, p. 180509, Nov 2017.

[2] S. Endo, S. C. Benjamin, and Y. Li, "Practical quantum error mitigation for near-future applications," *Phys. Rev. X*, vol. 8, p. 031027, Jul 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.8.031027

[3] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, and T. E. O'Brien, "Quantum Error Mitigation," Oct. 2022.

[4] S. Zhang, Y. Lu, K. Zhang, W. Chen, Y. Li, J.-N. Zhang, and K. Kim, "Error-mitigated quantum gates exceeding physical fidelities in a trapped-ion system," *Nature Communications*, vol. 11, no. 1, p. 587, 2020. [Online]. Available: https://doi.org/10.1038/s41467-020-14376-z

[5] A. Mari, N. Shammah, and W. J. Zeng, "Extending quantum probabilistic error cancellation by noise scaling," *Phys. Rev. A*, vol. 104, p. 052607, Nov 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.104.052607

[6] S. Ferracin, A. Hashim, J.-L. Ville, R. Naik, A. Carignan-Dugas, H. Qassim, A. Morvan, D. I. Santiago, I. Siddiqi, and J. J. Wallman, "Efficiently improving the performance of noisy quantum computers," Jul. 2022.

[7] D. Greenbaum, "Introduction to quantum gate set tomography," *arXiv preprint arXiv:1509.02921*, 2015.

[8] E. Nielsen, J. K. Gamble, K. Rudinger, T. Scholten, K. Young, and R. Blume-Kohout, "Gate Set Tomography," *Quantum*, vol. 5, p. 557, Oct. 2021. [Online]. Available: https://doi.org/10.22331/q-2021-10-05-557

[9] S. T. Flammia and J. J. Wallman, "Efficient estimation of Pauli channels," *ACM Transactions on Quantum Computing*, vol. 1, no. 1, pp. 1–32, 2020.

[10] E. van den Berg, Z. K. Minev, A. Kandala, and K. Temme, "Probabilistic error cancellation with sparse pauli-lindblad models on noisy quantum processors," 2022. [Online]. Available: https://arxiv.org/abs/2201.09866

[11] E. Nielsen, K. Rudinger, T. Proctor, A. Russo, K. Young, and R. Blume-Kohout, "Probing quantum processor performance with pygsti," *Quantum science and Technology*, vol. 5, no. 4, p. 044002, 2020.

[12] R. LaRose, A. Mari, S. Kaiser, P. J. Karalekas, A. A. Alves, P. Czarnik, M. El Mandouh, M. H. Gordon, Y. Hindy, A. Robertson, P. Thakre, M. Wahl, D. Samuel, R. Mistri, M. Tremblay, N. Gardner, N. T. Stemen, N. Shammah, and W. J. Zeng, "Mitiq: A software package for error mitigation on noisy quantum computers," *Quantum*, vol. 6, p. 774, Aug. 2022. [Online]. Available: https://doi.org/10.22331/q-2022-08-11-774

[13] B. McDonough, "benmcdonough20/autonomouspertools: v0.2.0-alpha," Oct. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7197234

[14] C. Song, J. Cui, H. Wang, J. Hao, H. Feng, and Y. Li, "Quantum computation with universal error mitigation on a superconducting quantum processor," *Science Adv.*, vol. 5, no. 9, 2019.

[15] R. Harper, S. T. Flammia, and J. J. Wallman, "Efficient learning of quantum noise," *Nat. Phys.*, vol. 16, no. 12, pp. 1184–1188, Dec. 2020.

[16] D. C. McKay, A. W. Cross, C. J. Wood, and J. M. Gambetta, "Correlated Randomized Benchmarking," Mar. 2020.

[17] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling," *Phys. Rev. A*, vol. 94, no. 5, p. 052325, 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.94.052325

[18] P. J. Karalekas, N. A. Tezak, E. C. Peterson, C. A. Ryan, M. P. da Silva, and R. S. Smith, "A quantum-classical cloud platform optimized for variational hybrid algorithms," *Quantum Sci. Tech.*, vol. 5, no. 2, p. 024003, apr 2020. [Online]. Available: https://doi.org/10.1088%2F2058-9565%2Fab7559

[19] E. van den Berg, Z. K. Minev, and K. Temme, "Model-free readout-error mitigation for quantum expectation values," *Phys. Rev. A*, vol. 105, p. 032620, Mar 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.105.032620

[20] M. Lostaglio and A. Ciani, "Error mitigation and quantum-assisted simulation in the error corrected regime," *Phys. Rev. Lett.*, vol. 127, p. 200506, Nov 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.127.200506

[21] C. Piveteau, D. Sutter, S. Bravyi, J. M. Gambetta, and K. Temme, "Error mitigation for universal gates on encoded qubits," *Phys. Rev. Lett.*, vol. 127, p. 200505, Nov 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.127.200505

[22] Y. Suzuki, S. Endo, K. Fujii, and Y. Tokunaga, "Quantum error mitigation as a universal error reduction technique: Applications from the nisq to the fault-tolerant quantum computing eras," *PRX Quantum*, vol. 3, p. 010345, Mar 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.3.010345