Data-driven adaptive quantum error mitigation for probability distribution

Rion Shimazu,[1] Suguru Endo,[1,2] Shigeo Hakkaku,[1,2] and Shinobu Saito[1,*]

[1] *NTT Computer and Data Science Laboratories, NTT Inc., Musashino 180-8585, Japan*
[2] *NTT Research Center for Theoretical Quantum Information, NTT Inc., 3-1 Morinosato Wakanomiya, Atsugi, Kanagawa, 243-0198, Japan*

Quantum error mitigation (QEM) has been proposed as a class of hardware-friendly error suppression techniques. While QEM has been primarily studied for mitigating errors in the estimation of expectation values of observables, recent works have explored its application to estimating noiseless probability distributions. In this work, we propose two protocols to improve the accuracy of QEM for probability distributions, inspired by techniques in software engineering. The first is the N-version programming method, which compares probability distributions obtained via different QEM strategies and excludes the outlier distribution, certifying the feasibility of the error-mitigated distributions. The second is a consistency-based method for selecting an appropriate extrapolation strategy. Specifically, we prepare K data points at different error rates, choose $L < K$ of them for extrapolation, and evaluate error-mitigated results for all $\binom{K}{L}$ possible choices. We then select the extrapolation method that yields the smallest variance in the error-mitigated results. This procedure can also be applied bitstring-wise, enabling adaptive error mitigation for each probability in the distribution.

I. INTRODUCTION

Near-term quantum computing with noisy intermediate-scale quantum (NISQ) or early fault-tolerant quantum computers is error-prone due to the limited functionality of quantum error correction (QEC) [1–5]. Due to its hardware efficiency, quantum error mitigation (QEM) is an alternative to QEC [6–8]. QEM post-processes the measurement outcomes from quantum computers to estimate the error-mitigated result at the cost of more repetitions on quantum computers.

QEM has been considered a powerful tool for high-accuracy quantum computing, and various QEM methods have been proposed so far. For example, zero-noise extrapolation (ZNE) estimates the error-free result by extrapolating from erroneous and error-boosted results [8–10]. Although the ZNE method is hardware-efficient and has shown significant experimental success in improving computational accuracy [11], it remains a heuristic approach without a formal guarantee of accuracy. Probabilistic error cancellation (PEC) tries to cancel the effect of noise by utilizing characterized noise information obtained in advance [8, 9]. However, noise characterization can never be perfect, which causes an inevitable bias in error-mitigated results. Since the accuracy and bias of these QEM methods are unknown in general, there is no established way for QEM users to select the most suitable QEM method for their use cases. This poses the risk that they may inadvertently employ an unsuitable QEM method with severely limited accuracy.

Traditionally, QEM techniques have primarily been employed to estimate the expectation values of observables computed from the output of quantum circuits [6, 7]. However, recent research has begun to explore QEM methods aimed at reconstructing the *entire output distribution* of the output of noisy quantum circuits [12, 13]. For instance, the method proposed in Ref. [12] takes this approach by estimating the error-mitigated output distribution based on the noisy measurement data, and then sampling from the reconstructed distribution. The authors of Ref. [12] benchmarked their method on a toy model of the quantum phase estimation task [14], demonstrating its applicability and discussing its potential advantages. Notably, such QEM techniques targeting output distributions are not limited to quantum phase estimation, but are also applicable to other distribution-based quantum tasks such as quantum-selected configuration interaction [15]. In fact, QEM was employed in the experiments of the quantum-selected configuration interaction, as reported in Ref. [16]. Therefore, the methods proposed in Refs. [9, 11] represent a significant broadening of the scope of QEMs. However, as with conventional QEM approaches, selecting an appropriate error mitigation strategy remains a key challenge in practice.

Here, we propose two methods to systematically and adaptively select the feasible QEM strategy, mainly for the error-mitigated probability distribution. The first method is inspired by the N-version programming used in the classical software engineering field [17]. The conceptual figure is shown in Fig. 1. For instance, in Ref. [18], the N-version of programming certifies the computation outcomes by comparing the probability distributions obtained from similar computations, e.g., probability distributions corresponding to solving the same problem using different algorithms. Because we can prepare several error-mitigated distributions through Ref. [12] via different applications of QEM strategies, we can compare these error-mitigated distributions with each other and choose the accurate probability distribution based on certain metrics. Here, we select the probability distribution that has the smallest total variational distance (TVD) to

* shinobu.saito@ntt.com

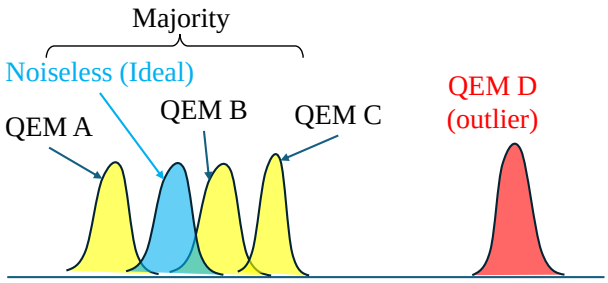

FIG. 1. The N-version programming approach calculates the distances among the distributions of the given QEM methods (QEMs A to D in this figure) and effectively identifies the method that deviates most from the majority (in this case, QEM D).

other distributions, because a large TVD indicates that the probability distribution is highly likely to be atypical.

The second method selects a suitable QEM method by comparing the output obtained using different QEM strategies to check the consistency of each QEM strategy. This method is also inspired by MAPE-K, commonly used to construct self-adaptive software systems [19]. MAPE-K defines four functions (*M*onitor, *A*nalyze, *P*lan, *E*xecute) and a *K*nowledge base that manages data shared between them. It enables systems to adapt to changing conditions and maintain optimal performance. By employing the concept of MAPE-K (i.e., self-adaptive), we can plan and execute the selection of the most suitable QEM method through monitoring and analysis of multiple different error rates (i.e., changing conditions). The conceptual figure for our protocol is shown in Fig. 2. First, we prepare data points for several error rates, choosing only a subset for extrapolation. We then compare the results of different extrapolation strategies for all possible combinations of the chosen data points. Next, we select the extrapolation strategy that yields the lowest variance in the error-mitigated outcomes. While we can perform this method to improve the estimation of the single-observable expectation value, we can apply this method to each measurement probability for the computation basis measurement, improving the accuracy of the overall error-mitigated probability distribution.

We verify the performance of our methods through numerical simulations using a Trotter-based Hamiltonian simulation algorithm for a transverse Ising Hamiltonian. For the N-version programming method, we compare the error-mitigated distributions obtained from extrapolation strategy including that provided by the Mitiq [20] QEM software package, and confirmed that we can systematically exclude the error-mitigated distribution that deviates significantly from others and the ideal distribution. In addition, when applying the consistency method to each bin of the probability distribution, it selects the most suitable QEM strategy in the dominant cases, yielding a naturally error-mitigated distribution with the smallest distance from the ideal distribution.

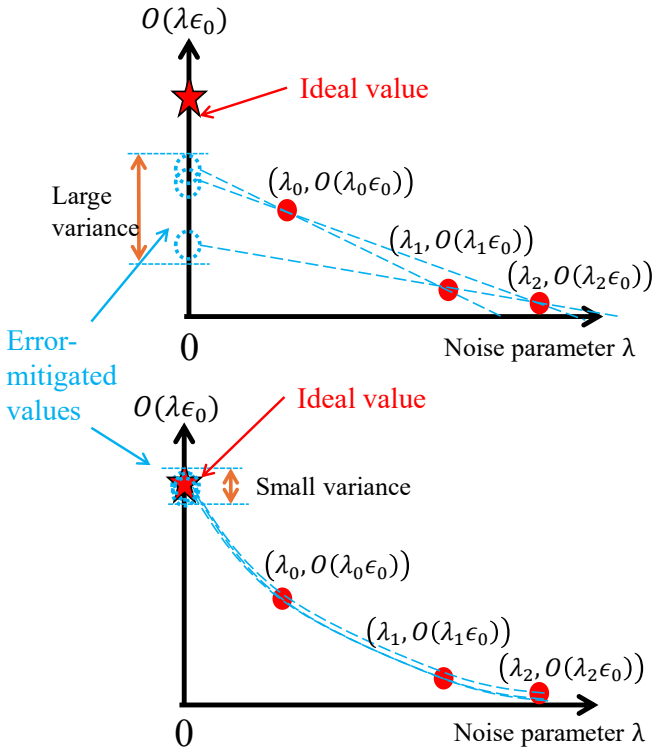

FIG. 2. The consistency-based method. In this approach, we hypothesize that when the optimal QEM method is employed, the mitigation results remain almost unchanged regardless of which data points are used. Under this assumption, the method that yields the smallest variance in QEM results across different data points is regarded as the most "consistent", and thus identified as the optimal one.

II. QUANTUM ERROR MITIGATION FOR PROBABILITY DISTRIBUTION

Here, we describe the recently introduced quantum error mitigation for probability distribution presented in Ref. [12]. Note that the expectation value of each POVM element $\Pi_z = |z\rangle\langle z|$ is equivalent to the probability p_z of obtaining the bitstring z, i.e., $p_z = \mathrm{Tr}[\rho\Pi_z]$ for a density matrix ρ representing the quantum state. We can interpret this method as mitigating the expectation value of each POVM element. Ref. [12] mainly considers QEM methods using the linear ansatz [21]:

$$\rho_{\mathrm{QEM}} = \sum_k c_k \rho_k,$$

where $c_k \in \mathbb{R}$ and ρ_k is a noisy state. The error-mitigated state ρ_{QEM} obtained via extrapolation and probabilistic error cancellation can be explained with the linear ansatz QEM state. Then, each error-mitigated probabil-

ity p_z^{QEM} can be represented as

$$\begin{aligned} p_z^{\mathrm{QEM}} &= \mathrm{Tr}[\Pi_z \rho_{\mathrm{QEM}}] \\ &= \sum_k c_k \mathrm{Tr}[\Pi_z \rho_k] \\ &= \Gamma \sum_k p_k \mathrm{sgn}(c_k) \mathrm{Tr}[\Pi_z \rho_k], \end{aligned} \tag{1}$$

where $\Gamma = \sum_k |c_k|$ and $p_k = |c_k|/\Gamma$. The third line in Eq. (1) allows for the Monte-Carlo implementation of the error-mitigated probability p_z^{QEM}. We randomly generate the state ρ_k with probability p_k and measure the state with the computational basis. We define the random variable

$$\hat{\mu}_z = \mathrm{sgn}(c_k)\hat{m}_z,$$

where $\hat{m}_z = 1$ when the measured result is z and otherwise 0. Then, we can show

$$p_z^{\mathrm{QEM}} = \Gamma \mathbb{E}[\hat{\mu}_z].$$

Because of $[\Pi_z, \Pi_{z'}] = 0 \ \forall z, z'$, the computational basis measurement allows for the simultaneous estimation of the overall probability distribution $\{p_z\}_z$, i.e., when $\hat{m}_z = 1$, $\hat{m}_{z'} = 0$ for $\forall z' \neq z$. Then, Ref. [12] shows

$$\sum_z \mathrm{Var}[p_z^{\mathrm{QEM}}] \leq \frac{\Gamma^2}{N_{\mathrm{meas}}}$$

for the number of measurements N_{meas}. This indicates that the sampling overhead of the probability distribution estimation is comparable to that of a single-observable estimation.

III. QUANTUM ERROR MITIGATION FOR PROBABILITY DISTRIBUTION WITHOUT MONTE-CARLO AND BEYOND LINEAR ANSATZ

However, the Monte-Carlo implementation of QEM methods is not hardware-friendly because it requires changing the structure of the quantum circuit for each execution. Here, we consider simply estimating the error-mitigated probability p_z^{QEM} as in the first line of Eq. (1), i.e., $p_z^{\mathrm{QEM}} = \sum_k c_k \mathrm{Tr}[\Pi_z \rho_k] = \sum_k c_k p_z^{(k)}$. In this case, the variance of the error-mitigated probability can be described as

$$\begin{aligned} \mathrm{Var}[p_z^{\mathrm{QEM}}] &= \sum_k c_k^2 \mathrm{Var}[p_z^{(k)}] \\ &= \sum_k c_k^2 \frac{p_z^{(k)} - p_z^{(k)2}}{N_{\mathrm{meas}}^{(k)}}, \end{aligned}$$

which results in

$$\begin{aligned} \sum_z \mathrm{Var}[p_z^{\mathrm{QEM}}] &= \sum_k c_k^2 \frac{1 - \sum_z p_z^{(k)2}}{N_{\mathrm{meas}}^{(k)}} \\ &\leq \sum_k \frac{c_k^2}{N_{\mathrm{meas}}^{(k)}}, \end{aligned}$$

where $N_{\mathrm{meas}}^{(k)}$ is the number of samples assigned to measure the noisy state ρ_k, satisfying $\sum_k N_{\mathrm{meas}}^{(k)} = N_{\mathrm{meas}}$.

Compared with the Monte-Carlo implementation, the sampling overhead generally becomes worse as $\sum_k c_k^2/N_{\mathrm{meas}}^{(k)} \geq (\sum_k |c_k|)^2/N_{\mathrm{meas}}$. Note that, by optimizing the sample number allocation, the two sampling overheads become equivalent. Therefore, when the number of erroneous states in the linear ansatz in Eq. (1) is not significant; the straightforward QEM strategy works as an alternative method to the Monte-Carlo method.

Furthermore, when we do not resort to the Monte-Carlo implementation, we are not restricted to the linear ansatz in Eq. (1). For example, while the exponential extrapolation method cannot be described by the linear ansatz, it shows excellent performance in actual large-scale experiments. Although the exponential extrapolation method is not applicable when $p_z = 0$ due to division, it is desirable to incorporate this method when possible.

IV. PROPOSED METHODS

Many QEM methods have been proposed, but the accuracy of those methods varies significantly depending on factors such as the noise characteristics of quantum computers or the structure of quantum circuits. Therefore, it is important to select the method that maximizes accuracy according to the specific use case. However, selecting the best method is difficult because the effect of noise characteristics and circuit structure on the accuracy of QEMs is unknown in general. This may make QEM users choose an unsuitable QEM method for their use cases, and users would be left with inaccurate results of QEMs. To overcome the challenge, we propose two methods that utilize multiple QEM methods to produce high-accuracy results from quantum computers: the N-version programming method and the consistency-based method.

A. N-version programming method

The first method is the N-version programming method. N-version programming is a concept that is frequently employed in the field of software engineering to obtain an error-free result [17]. The basic concept involves solving the same problem with different setups, such as various algorithms and environments, and then comparing the outcomes from each setup to extract the feasible ones. The celebrated example of the N-version of programming is to compute the distance of each probability distribution and reject the distribution with the largest distance from other distributions to avoid atypical results [18].

Here, we propose applying the N-version programming methods to the error-mitigated probability distributions. The schematic figure of this method is shown in Fig. 1. In

our case, the "N-versions" are obtained through the applications of different QEM strategies. For example, we prepare three types of error-mitigated probability distributions via linear extrapolation, second-order Richardson extrapolation, and exponential extrapolation from noisy probability distributions. For extrapolation QEM methods, we can prepare the "N-versions" only via classical processing of noisy probability distributions prepared in advance. After we prepare N instances of error-mitigated probability distributions, we compute the TVD for all the $\binom{N}{2}$ pairs of error-mitigated probability distributions. Then, denoting each error-mitigated distribution $\{q_k\}_{k=1}^{N_{\mathrm{QEM}}}$ with N_{QEM} being the number of error-mitigated distributions, we compute TVDs $D(q_k, q'_k)$ for $\forall k, k'$, and select k such that $\sum_{k'=1}^{N_{\mathrm{QEM}}} D(q_k, q'_k)$ is minimized.

B. Consistency-based method

In the second method, we select the most consistent QEM method by comparing outcomes from different extrapolation QEM methods. The conceptual figure is shown in Fig. 2. We first prepare the noisy expectation values for K different noise parameters, denoted as $\{\langle O(\lambda_k \varepsilon_0)\rangle\}_{k=0}^{K-1}$ for noise stretch factor $1 = \lambda_0 < \lambda_1 < \lambda_2 ... < \lambda_{K-1}$ and the original error rate ε_0. Then, we select $L < K$ data points to apply the L-point extrapolation method. Then, by performing the extrapolation for $\binom{K}{L}$ combinations of erroneous data points, we can prepare $\beta = \binom{K}{L}$ error-mitigated values, denoted as $\{a_j\}_{j=1}^{\beta}$. Then we compute the variance of the error-mitigated values $\{a_j\}_{j=1}^{\beta}$. Note that a small variance of the error-mitigated values indicates that the extrapolation strategy yields a consistent error-mitigated result. We perform the same procedures for other extrapolation strategies, and choose the QEM strategy with the smallest variance as the most consistent extrapolation curve. While the extrapolation method is inherently heuristic and generally lacks a theoretical guarantee of accuracy, our approach enables us to establish a certain degree of performance guarantee. We can also apply this method in a bit-string-wise manner to each probability distribution $\{p_z^{\mathrm{QEM}}\}_z$, e.g., we apply a linear extrapolation for $z = 01$ and an exponential extrapolation for $z = 10$, which is determined by the consistency evaluation.

V. NUMERICAL EXPERIMENTS

A. Experimental setup and procedure

For each of the two proposed methods, we conduct numerical experiments using a quantum circuit simulator. In our numerical experiments, we consider the time evolution of a physical model described by the following one-dimensional transverse-field Ising Hamiltonian:

$$H_{\mathrm{TFI}} = J \sum_{j=1}^{N_Q} Z_j Z_{j+1} + B \sum_{j=1}^{N_Q} X_j,$$

where $Z_i(X_i)$ is the Pauli Z(X) operator on the i-th site. For this Hamiltonian, the time-evolution operator over a period t is expressed as $e^{-iH_{\mathrm{TFI}}t}$. Here, in order to represent this time-evolution operator as a quantum circuit, we perform the following first-order Trotterization:

$$e^{-iH_{\mathrm{TFI}}t} \approx U_T,$$

where

$$U_T \equiv \left\{ \exp\left(i \sum_{j=1}^{N_Q} Z_j Z_{j+1} \frac{t}{M} \right) \times \exp\left(-i \sum_{j=1}^{N_Q} X_i \frac{t}{M} \right) \right\}^M,$$

with M the Trotter number. See Sec. A for the details of the first-order Trotterization. For the simulation of the circuit U_T, we employ Qiskit [22]. Figure 3 shows an example of a 3-qubit Trotterized circuit implementing the unitary transformation U_T. The circuit U_T can be expressed using the rotation gate and CNOT gate classes provided by Qiskit. The simulations are performed using the backend named `FakeMarrakesh`, which simulates the noise characteristics of IBM's quantum hardware. Other software used in the simulations is summarized in Tab. III in Sec. B.

Parameter	Description	Value(s)
N_Q	Numer of qubits	=10
N_{meas}	Number of measurements for each circuit	=5000
t	Duration of the time evolution	=1
J	Coefficients of the Hamiltonian	$\in \{1, 2, \cdots, 10\}$
B		
λ	Scale factor	$\in \{3, 5\}$

TABLE I. The common parameters used in the experiments.

The parameters used in our experiments are shown in Tab. I. In this numerical experiment, the coefficients J and B of the Hamiltonian H_{TFI} are varied from 1 to 10, resulting in a total of $10 \times 10 = 100$ combinations of (J, B). For each pair of J and B, we first simulate the unitary operation U_T and compute both the output distribution $\{p_z\}_z$. Subsequently, we construct a modified circuit $U_{T,\lambda}$ whose error is amplified by a scale factor λ, and simulate the corresponding output distribution $\{p_{z,\lambda}\}_z$. Note that these distributions are later used for performing quantum error mitigation. The procedure for amplifying the error is presented in Sec. C.

After obtaining the distributions $\{p_z\}_z$ and $\{p_{z,\lambda}\}_{z,\lambda}$, we apply the QEM methods and the proposed methods using these results and obtain the error-mitigated

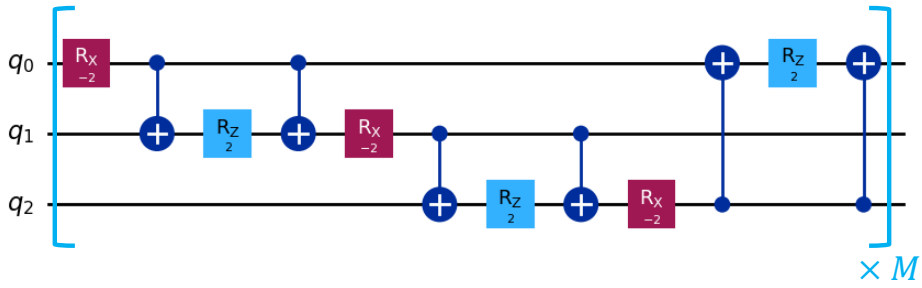

FIG. 3. An example of a 3-qubit Trotterized circuit for the one-dimensional transverse-field Ising model. A single block composed of RX, RZ, and CNOT gates is repeatedly applied for M times. We used Qiskit [22] to draw the circuit.

distributions $\{p_z^{\mathrm{QEM}}\}_z$. In this experiment, we adopt extrapolation-based QEM methods [7, 23]. Specifically, we employ linear extrapolation, second-order Richardson extrapolation, exponential extrapolation, and polynomial-exponential extrapolation methods. The first three methods estimate the error-mitigated probability distribution $\{p_z^{\mathrm{QEM}}\}_z$ according to the following equations:

Linear:

$$p_z^{\mathrm{QEM}} = \frac{3p_z - p_{z,3}}{2}.$$

Second-order Richardson:

$$p_z^{\mathrm{QEM}} = \sum_{\lambda\in\{1,3,5\}} C_\lambda p_{z,\lambda},$$

where $p_{z,1} = p_z$ and

$$C_\lambda = \prod_{\lambda,\lambda'\in\{1,3,5\},\ \lambda\neq\lambda'} \frac{\lambda'}{\lambda'-\lambda}.$$

Exponential:

$$p_z^{\mathrm{QEM}} = (p_z)^{3/2}(p_{z,3})^{-1/2}.$$

In the fourth method, p_z^{QEM} is estimated numerically via nonlinear regression for the polynomial-exponential extrapolation function, rather than by an analytical formula. The employed nonlinear regression model is as follows:

Polynomial-Exponential Extrapolation:

$$p_z^{\mathrm{QEM}} = \theta_0 \exp\left(\theta_1\lambda + \theta_2\lambda^2\right).$$

Here, $\theta_0, \theta_1, \theta_2 \in \mathbb{R}$ are the parameters determined by the numerical fitting. This model is among those supported by the Mitiq package [20]. We employ Mitiq's `PolyExpFactory` class to perform the numerical fitting.

Note that, regardless of the QEM method, the QEM procedure is applied independently for each z only if p_z or any of the values in the set $\{p_{z,\lambda}\}_{z,\lambda}$ is nonzero. If p_z and all values in $\{p_{z,\lambda}\}_{z,\lambda}$ are zero for a given z, the corresponding error-mitigated probability p_z^{QEM} is set to zero. Then, the TVD between the error-mitigated distributions generated by each QEM method and the ideal distribution is calculated, and the methods are ranked by their TVD values. The procedure described above is performed for all combinations of J and B, with each taking values from 1 to 10, yielding a total of 100 runs.

B. Numerical results

Here, we present the results of the experiments conducted for the proposed methods. Figure 4 summarizes the ranking of the QEM methods selected by the N-version programming approach in terms of TVD. The Trotter number M was varied from 5 to 10. For each fixed M, 100 trials are conducted, and the rank of the QEM method chosen by the proposed approach, based on TVDs among all candidate methods, is recorded. The vertical axis indicates the number of occurrences for each rank. In this experiment, the QEM method most frequently selected by the proposed approach is

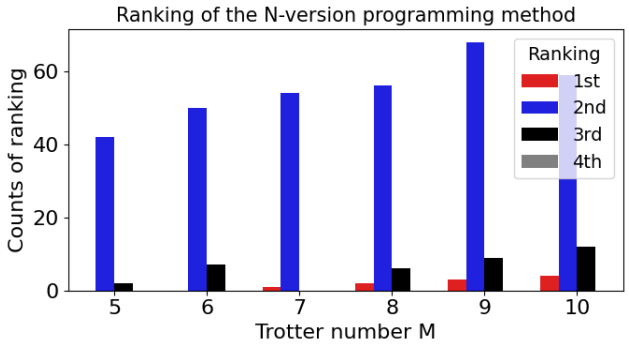

FIG. 4. Count of the N-version-programming method's ranks among QEM methods sorted by the corresponding TVDs. The N-version programming method is never ranked 4th for any parameter M, indicating that the method identifies the outlier as depicted in Fig. 1.

Method	No. of 1st places	No. of 2nd places	No. of 3rd places
Linear	1	4	58
Richardson	0	0	12
Exponential	22	54	2
Consistency-based method	60	21	1

TABLE II. Numerical experimental results of the consistency-based method in Sec. IV B. The columns "No. of 1st–4th places" present, for each method, the frequency with which it obtained each respective rank based on TVD across 10 × 10 = 100 experimental runs.

the second-ranked one, while the fourth-ranked (i.e., the worst-performing) method has never been chosen. This result indicates that, although the proposed method does not necessarily select the best-performing QEM, it consistently avoids poor (outlier) methods. This demonstrates that the anticipated behavior shown in Fig. 1 has been successfully observed. We remark that the method considered an outlier is always the polynomial-exponential extrapolation method.

Table II presents the results of the experiment for the consistency-based method with $M = 10$. In this numerical experiment, only linear, second-order Richardson, and exponential extrapolation are considered because the polynomial-exponential, which had been an outlier, can be excluded. The columns "No. of 1st–3rd places" show how often each method attains each respective rank, based on TVDs computed over 10×10 experimental runs. The consistency-based method achieves the first rank in 60 out of 100 trials, demonstrating that it is significantly more likely to achieve higher accuracy than the other QEM methods. Moreover, even when it does not rank first, our method ranks second in 20 trials and ranks last only once. In contrast, among the conventional QEM methods, specific approaches, such as the linear extrapolation method, perform considerably worse, ranking last in 58 trials.

VI. CONCLUSION AND DISCUSSIONS

In this work, we propose two methods to improve error-mitigation performance, inspired by software engineering for reliable classical computing. The first method is the N-version programming method, which compares the multiple error-mitigated distributions and excludes the atypical distribution. The second method is the consistency-based method. This method computes the variances of the error-mitigated results across multiple QEM methods and selects the result with the smallest variance as the most consistent; we also apply this method to each bin of the probability distribution. We confirm that the N-version programming method successfully excludes the most faulty error-mitigated distribution, and that the consistency-based method for probability distributions can achieve the most accurate error-mitigated distribution in the majority of cases.

While we focus on zero-noise extrapolation to demonstrate our protocol, our method can be naturally extended to other QEM methods, such as probabilistic error cancellation (PEC) [9]. It may be interesting to include the PEC or the PEC-extrapolation combination for obtaining the error-mitigated distribution [24, 25], and to examine how the performance of our method changes with the accuracy of the noise characterization required for the PEC implementation.

In addition, the application of this method in the error-corrected regime is worth exploring. While our protocol has been demonstrated in the NISQ setup, i.e., any QEC functionality is not assumed, the interplay between the quantum error correction/detection and QEM has been recently studied for high-accuracy quantum computing in the early fault-tolerant quantum computing era [26–30]. Therefore, delving further into the optimization of our method and seeking suitable software engineering methods that consider the QEC functionality is of practical interest.

ACKNOWLEDGMENTS

The authors would like to thank Takayoshi Shiraki, Masayuki Oda, and Hidetoshi Sawada for valuable discussions and support with numerical simulations.

[1] S. J. Devitt, W. J. Munro, and K. Nemoto, Rep. Prog. Phys. **76**, 076001 (2013).
[2] D. A. Lidar and T. A. Brun, *Quantum error correction* (Cambridge university press, 2013).
[3] J. Preskill, Quantum **2**, 79 (2018).
[4] J. Preskill, ACM Transactions on Quantum Computing **6**, 18 (2025).
[5] J. Eisert and J. Preskill, Mind the gaps: The fraught road to quantum advantage (2025), arXiv:2510.19928 [quant-ph].
[6] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, and T. E. O'Brien, Rev. Mod. Phys. **95**, 045005 (2023).
[7] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan, J. Phys. Soc. Jpn. **90**, 032001 (2021).
[8] S. Endo, S. C. Benjamin, and Y. Li, Phys. Rev. X **8**, 031027 (2018).
[9] K. Temme, S. Bravyi, and J. M. Gambetta, Phys. Rev. Lett. **119**, 180509 (2017).
[10] Y. Li and S. C. Benjamin, Phys. Rev. X **7**, 021050 (2017).

[11] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala, Nature **618**, 500 (2023).
[12] K. Liu and Z. Cai, Quantum error mitigation for sampling algorithms (2025), arXiv:2502.11285 [quant-ph].
[13] A. Muqeet, S. Ali, and P. Arcaini, IEEE Software **42**, 28 (2025).
[14] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).
[15] K. Kanno, M. Kohda, R. Imai, S. Koh, K. Mitarai, W. Mizukami, and Y. O. Nakagawa, Quantum-selected configuration interaction: classical diagonalization of hamiltonians in subspaces selected by quantum computers (2023), arXiv:2302.11320 [quant-ph].
[16] Y. O. Nakagawa, M. Kamoshita, W. Mizukami, S. Sudo, and Y.-y. Ohnishi, J. Chem. Theory Comput. **20**, 10817 (2024).
[17] L. Chen and A. Avizienis, in *Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995, ' Highlights from Twenty-Five Years'.* (1995) pp. 113–.
[18] S. Saito, S. Endo, and Y. Suzuki, in *2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW)* (2024) pp. 119–120.
[19] S. R. White, J. E. Hanson, I. Whalley, D. M. Chess, A. Segal, and J. O. Kephart, Integrated Computer-Aided Engineering **13**, 173 (2006).
[20] R. LaRose, A. Mari, S. Kaiser, P. J. Karalekas, A. A. Alves, P. Czarnik, M. E. 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, Quantum **6**, 774 (2022).
[21] Z. Cai, A practical framework for quantum error mitigation (2023), arXiv:2110.05389 [quant-ph].
[22] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, Quantum computing with Qiskit (2024), arXiv:2405.08810 [quant-ph].
[23] S. Endo, Q. Zhao, Y. Li, S. Benjamin, and X. Yuan, Phys. Rev. A **99**, 012334 (2019).
[24] A. Mari, N. Shammah, and W. J. Zeng, Phys. Rev. A **104**, 052607 (2021).
[25] J. Sun, X. Yuan, T. Tsunoda, V. Vedral, S. C. Benjamin, and S. Endo, Phys. Rev. Appl. **15**, 034026 (2021).
[26] Y. Suzuki, S. Endo, K. Fujii, and Y. Tokunaga, PRX Quantum **3**, 010345 (2022).
[27] C. Piveteau, D. Sutter, S. Bravyi, J. M. Gambetta, and K. Temme, Phys. Rev. Lett. **127**, 200505 (2021).
[28] M. Lostaglio and A. Ciani, Phys. Rev. Lett. **127**, 200506 (2021).
[29] K. Tsubouchi, Y. Suzuki, Y. Tokunaga, N. Yoshioka, and S. Endo, Phys. Rev. A **108**, 042426 (2023).
[30] Y. Xiong, D. Chandra, S. X. Ng, and L. Hanzo, IEEE Access **8**, 228967 (2020).
[31] M. Suzuki, Commun.Math. Phys. **51**, 183 (1976).
[32] S. Lloyd, Science **273**, 1073 (1996).
[33] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, Phys. Rev. X **11**, 011020 (2021).
[34] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (2020) pp. 306–316.

OS	Ubuntu 24.04 LTS
Python	3.12.9
pip	25.2
qiskit	1.4.0
qiskit-aer-gpu	0.15.1
qiskit-ibm-runtime	0.36.0
simulator backend	`FakeMarrakesh`
mitiq	0.47.0

TABLE III. Software used for our numerical experiments.

Appendix A: Trotterization

Here, we briefly explain the Trotterization algorithm [31–33], which approximates an exponential of a sum of operators, such as a time evolution operator, by a product of elementary exponentials. It is used as a subroutine of many quantum algorithms, including Hamiltonian simulation [32] and phase estimation [14]. In the following, we explain the first-order Trotterization algorithm using a time evolution operator as an example. Let $H = \sum_i^L H_i$, where each H_i acts on a constant number of qubits. The first-order Trotterization of the time evolution operator e^{-iHt} is given by

$$S_1(t) = \prod_j e^{-iH_j t}.$$

For the Hamiltonian simulation using a quantum circuit, we implement the first-order Trotterization $S_1(t/M)$ for M times. M is called the Trotter number, and it controls the algorithmic error (Trotter error). Ref. [32] has shown that

$$e^{-iHt} - \{S_1(t/M)\}^M = \frac{t^2}{2M}\sum_{i<j}[H_i, H_j] + \sum_k^\infty E_k,$$

where $\|E_k\| \leq M\|Ht/M\|^k/k!$ for the operator norm $\|\cdot\|$. If the quantum circuits are error-free, the algorithmic error can be arbitrarily suppressed as the Trotter number M increases.

Appendix B: Software used in numerical experiments

The software used in our numerical experiments is summarized in Tab. III.

Appendix C: Error amplification

In this section, we present the error-amplification procedure used in Sec. V A. The procedure is described in Procedure 1. For a given unitary circuit U, "Amplify-Error" effectively amplifies the error in the circuit by a scale factor λ. This is achieved by employing the gate

folding technique proposed in Ref. [34], in which redundant gates are inserted before and after each original gate to effectively increase the circuit's error rate by the factor λ. However, in our implementation, we perform an additional post-processing step to remove all $(\mathrm{SX})^\dagger$ gates after applying gate folding. This modification is necessary because, when the circuit shown in Fig. 3 is transpiled for the `FakeMarrakesh` backend, SX gates appear as part of the compiled circuit. If gate folding is applied to such a circuit, the resulting folded circuit inevitably contains $(\mathrm{SX})^\dagger$ gates. Since the `FakeMarrakesh` backend does not support the $(\mathrm{SX})^\dagger$ gate as a native operation, circuits containing this gate cannot be executed directly. Therefore, we removed all $(\mathrm{SX})^\dagger$ gates from the folded circuits before execution. As a consequence of this removal, the "AmplifyError" procedure sacrifices accuracy in the effective error-rate scaling.

Procedure 1 Error amplification via gate folding.

procedure AMPLIFYERROR(λ, U)
  ∘ Assume λ is an odd integer.
  ∘ U_λ : Empty circuit.
  ∘ g^{list} : List of the gates in U (excluding the measurement gates).
  for g in g^{list} **do**
    ∘ $U_\lambda \leftarrow (gg^\dagger)^{\lambda-1} g U_\lambda$.
  end for
  ∘ Remove the $(\mathrm{SX})^\dagger$ gates in U_λ. This is due to the constraints in the `FakeMarrakesh` backend, which does not support the $(\mathrm{SX})^\dagger$ gates.
return U_λ.
end procedure