QuFEM: Fast and Accurate Quantum Readout Calibration Using the Finite Element Method

Siwei Tan
Zhejiang University
Hangzhou, China
siweitan@zju.edu.cn

Liqiang Lu*
Zhejiang University
Hangzhou, China
liqianglu@zju.edu.cn

Hanyu Zhang
Zhejiang University
Hangzhou, China
hyzz@zju.edu.cn

Jia Yu
Zhejiang University
Hangzhou, China
yujia_cs@zju.edu.cn

Congliang Lang
Zhejiang University
Hangzhou, China
langcongliang@zju.edu.cn

Yongheng Shang
Zhejiang University
Hangzhou, China
yh_shang@zju.edu.cn

Xinkui Zhao
Zhejiang University
Hangzhou, China
zhaoxinkui@zju.edu.cn

Mingshuai Chen
Zhejiang University
Hangzhou, China
m.chen@zju.edu.cn

Yun Liang
Peking University
Beijing, China
ericlyun@pku.edu.cn

Jianwei Yin*
Zhejiang University
Hangzhou, China
zjuyjw@zju.edu.cn

Abstract

Quantum readout noise turns out to be the most significant source of error, which greatly affects the measurement fidelity. Matrix-based calibration has been demonstrated to be effective in various quantum platforms. However, existing methodologies are fundamentally limited in either scalability or accuracy. Inspired by the classical finite element method (FEM), a formal method to model the complex interaction between elements, we present our calibration framework named QuFEM. First, we apply a divide-and-conquer strategy that formulates the calibration as a series of tensor products with noise matrices. This matrices are iteratively characterized together with the calibrated probability distribution, aiming to capture the inherent locality of qubit interactions. Then, to accelerate the end-to-end calibration, we propose a sparse tensor-product engine to exploit the sparsity in the intermediate values. Our experiments show that QuFEM achieves 2.5×10^3× speedup in the 136-qubit calibration compared to the state-of-the-art matrix-based calibration technique [50], and provides 1.2× and 1.4× fidelity improvement on the 18-qubit and 36-qubit real-world quantum devices.

CCS Concepts: • **Hardware → Quantum technologies**.

*Corresponding Author

ASPLOS '24, April 27-May 1, 2024, La Jolla, CA, USA

ACM ISBN 979-8-4007-0385-0/24/04.
https://doi.org/10.1145/3620665.3640380

Keywords: quantum computing, readout calibration, quantum error mitigation

ACM Reference Format:
Siwei Tan, Liqiang Lu, Hanyu Zhang, Jia Yu, Congliang Lang, Yongheng Shang, Xinkui Zhao, Mingshuai Chen, Yun Liang, and Jianwei Yin. 2024. QuFEM: Fast and Accurate Quantum Readout Calibration Using the Finite Element Method. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24), April 27-May 1, 2024, La Jolla, CA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3620665.3640380

1 Introduction

The paradigm of quantum computing exhibits a high potential to outperform classical computing in solving complex problems, e.g., physical simulation [11], combinatorial optimization [4, 17], and many instances from the realm of artificial intelligence [28, 34]. When solving these problems on real-world quantum hardware, the computation inevitably encounters various sources of error [12, 13, 35, 60], amongst which, the readout error – noise incurred by reading the output from quantum devices and transforming it to classical data – has been demonstrated as the most significant source of error. This error amounts to 1%-10% per qubit operation [14, 43, 52] for most physical quantum implementations, thus drastically hindering the application of quantum computing.

The fidelity of readout can be improved by hardware-level optimization techniques, including frequency fine-tuning [56] and sapphire-based processor fabrication [3, 27, 59]. These techniques make the states of qubits more distinguishable by isolating them from external noises. Whereas a more cost-efficient approach is to apply software-level optimization, such as matrix-based calibration [45, 50], machine learning [29, 33], domain-specific patterns [53, 54], and Bayesian

estimation [14]. For readout that yields a probability distribution, the matrix-based calibration method is experimentally shown effective in superconducting [31], ion-trap [10], and optical [63] quantum hardware, which is hence widely-used in commercial quantum cloud platforms like IBMQ [23] and Rigetti [47]. This approach mainly consists of two steps: matrix characterization and matrix-vector multiplication (MVM). The characterization step profiles matrix elements by running benchmarking circuits on the quantum device with various initial states (for a noise-free quantum device, this step yields an identity matrix). The MVM step multiplies the resulting matrix with a probability vector of the measured probability distribution derived from the raw data after readout. The resultant probability vector is thereby calibrated with higher fidelity.

Although the matrix-based calibration can be readily deployed on the software stack, it is confronted with the trade-off between scalability and calibration accuracy. In order to maximize the accuracy, i.e., to precisely characterize the matrix, it requires exhaustively executing a tremendous number of benchmarking circuits to cover all possible measured outputs, which is exponential in the number of qubits [9, 21, 45, 50]. For instance, obtaining such a matrix for 16 qubits requires the execution of 6.5×10^4 (2^{16}) circuits, taking roughly 18.2 hours on the IBMQ Guadalupe quantum device. Beyond the matrix characterization, the MVM step is also computationally overwhelmed as the size of the matrix is exponential in the number of qubits. As an example, M3 [37] – a calibration method recently developed by IBMQ – exploits a pruning strategy based on a threshold of Hamming distances. However, it still requires 1.9PB memory and more than a year to conduct the MVM step for a 45-qubit quantum circuit output.

An orthogonal line of research aims at improving the scalability of the calibration process at the cost of reduced accuracy. Based on the observation that the non-zero elements of the calibration matrix distribute near its diagonal, several approaches employ the idea to characterize the matrix with fewer benchmarking circuits [9, 37, 50]. For instance, both IBU [50] and Yang et al. [61] apply a qubit-independent calibration matrix – calculated by the tensor product of a series of 2×2 meta-matrices – and thereby can scale to 81-qubit circuits. These approaches, however, fail to model the crosstalk between qubits, thus decreasing the readout fidelity. For example, on the 7-qubit IBMQ Perth quantum computer, excluding the crosstalk noise induces a reduction of the fidelity of the GHZ circuit [20] from 94.80% to 87.21%. Alternative techniques leverage the sparsity of the calibration matrix to alleviate the high computational complexity [37, 38]. Nonetheless, by pruning away elements that are no larger than 10^{-3} (accounting for 87.26%), the fidelity of the GHZ circuit reduces from 94.80% to 86.88%.

In this work, we propose QuFEM. The key novelty of it is to extend classical FEM to a new formulation of the calibration process. FEM serves as a popular technique in system studies that feature complex interactions between elements, such as aerodynamics for weather prediction [1] and thermodynamics [49] for flight vehicle design. A main feature of FEM is to partition the object into multiple pieces and iteratively analyze the evolution of each piece independently, which greatly reduces the analysis complexity meanwhile keeps the analysis accurate. The state of the overall system is gathered from these pieces. Readout calibration is, in essence, a simulation that inverses the evolution caused by qubit interactions, making it naturally suitable to be modeled by FEM.

QuFEM also applies an iterative calibration to the readout output, which reformulates the calibration as a series of tensor-product with multiple sub-noise matrices. These sub-matrices try to capture interactions between groups of qubits. The objective of the grouping scheme includes *1)* maximizing the locality within the group and *2)* minimizing the interaction between different groups, which aims to comprehensively cover more architectural details between qubits, making QuFEM closer to the golden calibration.

To quickly and accurately characterize these sub-matrices, we propose a generation approach that minimizes the number of benchmarking circuits. To be specific, QuFEM tries to identify the critical circuits that exhibit high interactions in some local regions. Considering the foundation that the interaction is mainly static and sparse on the physical device, the generation ends until the accuracy of the characterization is kept at a stable value. On the other hand, to accelerate the computation of calibration, i.e., a series of tensor-products, we develop a sparse tensor-product engine. Different from prior works that directly conduct pruning on the noise matrix, our method prunes the intermediate values to ensure higher characterization accuracy. In this manner, our sparse engine only shows polynomial complexity, leading to a huge reduction compared to prior works with exponential complexity. In summary, QuFEM achieves a high speedup for end-to-end calibration, meanwhile improving the fidelity of measurement operations.

The main contributions of this work are as follows:

- We propose QuFEM, a framework for calibrating the measurement error using FEM. Our reformulation features an iterative process that provides a deeper understanding of the characterization and the calibration.
- We propose an approach to minimize the benchmarking circuits for noise matrix characterization. Instead of exhaustive characterization in an exponential time complexity $\mathcal{O}(2^n)$. Our generation technique greatly reduces the characterization time to polynomial complexity $\mathcal{O}(n^2)$.
- We propose a sparse computation engine to accelerate the end-to-end calibration. Using our engine, the calibration time is only cubic to the number of qubits $\mathcal{O}(n^3)$, so does the memory requirement.

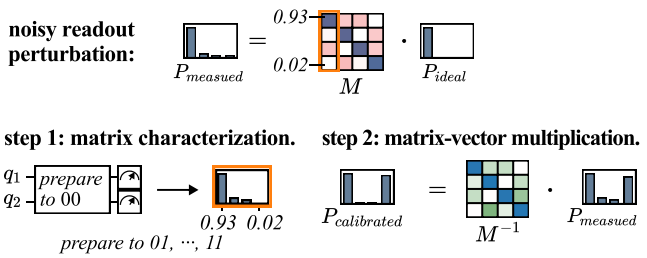


Figure 1. Readout calibration using noise matrix.

In Section 5, we theoretically prove that QuFEM has no more than polynomial complexity with respect to the number of qubits. In Section 6, we collect a dataset with 25M real machine samples on across 4 quantum cloud platforms, ranging from 7-qubit to 136-qubit quantum devices. The result suggests that QuFEM achieves $2.5{\times}10^3\times$ speedup in the 136-qubit calibration compared to the state-of-the-art matrix-based calibration technique [50] and provides 1.2× and 1.4× fidelity improvement on the 7-qubit and 18-qubit real-world quantum devices. The source code and dataset of QuFEM are publicly available on (https://github.com/JanusQ/QuFEM).

2 Background

2.1 Quantum Readout

Quantum readout is an operation to read the information from quantum bits (qubits) to classical bits. For most quantum algorithms, their outputs are probability distributions of bit strings obtained by executing the circuits multiple times. The bit string is in a $\{0,1\}^{N_q}$ space, where N_q is the number of qubits. The observed probability of a certain bit string represents its amplitude in the measured quantum superposition state. At the physical level, the readout is implemented by probing the qubit system and estimating the quantum state via physical quantity. For example, on the superconducting transmon qubits [6], the readout operation is realized by coupling the frequency of the qubit ω_q and the frequency of readout resonator ω_r. For each individual qubit, its state is classified by the frequency shift detected in the readout resonator:

$$\Delta\omega_r = g^2/|\omega_q - \omega_r| \tag{1}$$

where g is a constant coupling strength [25]. By comparing the drift with a threshold, we can discriminate whether each bit is 0 or 1 in the bit string. The sources of the readout errors include the imperfect operations [41], crosstalk [3, 24], and environmental noise [32, 32]. An important error is qubit interactions (e.g., crosstalk), which means the frequencies of other qubits leak to the measured qubit, leading to imperfections in the frequency shift.

2.2 Matrix-based Calibration

Mathematically, readout error leads to perturbations in the probability distribution. Such perturbation is formulated as a linear transformation from the ideal distribution P_{ideal} to the measured distribution $P_{measured}$:

$$P_{measured} = MP_{ideal}, \tag{2}$$

where M is defined as the *noise matrix* with the size of $2^{N_q} \times 2^{N_q}$. Theoretically, the noise matrix can be characterized by running benchmarking circuits with every basis state as input. As shown in the **step 1** of Figure 1, 2^{N_q} benchmarking circuits are executed, preparing all possible basis states (no superposition). Similar to prior works [9, 21, 45, 50], here we assume that the initial state is the ideal state, as it is prepared by high-fidelity single-qubit gates. The noise matrix is filled according to the output probability distribution after execution:

$$M[x][y] = P(measure = x \mid prepare = y). \tag{3}$$

For example, in Figure 1, 4 (2^2) circuits are executed for a 2-qubit system. The element in (3,0) is 0.02, which means that when the ideal state is $|00\rangle$, the probability of observing $|11\rangle$ is 0.02.

$$M[3][0] = P(measure = 11 \,|prepare = 00) = 0.02.$$

After getting the noise matrix M, we can calibrate any measured probability distribution from this quantum device using its inverse M^{-1}, namely *calibration matrix*. The calibration is conducted via an MVM operator between the measured distribution and the calibration matrix:

$$P_{calibrated} = M^{-1}P_{measured}. \tag{4}$$

The **step 2** in Figure 1 is an example of a 2-qubit calibration.

Table 1. Comparsion of 5 readout calibration techniques. "Poly." or "Exp." means the technique has polynomial or exponential complexity, respectively.

Methodology		Real	Qubit-independent	Sparsity-aware		FEM
Name			IBU [50] Yang et al. [61]	M3 [37]	Nation et al. [38]	QuFEM
Formulation		-	$\otimes\otimes$			$(\otimes)\cdot(\otimes)$
Noise matrix						
Scalability	Max qubit	20	80	45	14	≥500
	Charac. time (one-shot)	Exp. (×)	Linear (✓)	Exp. (×)	Exp. (×)	Linear (✓)
	MVM time	Exp.	Exp.	Exp.	Exp.	Poly.
Accuracy	Generality	All	GHZ, BV	All	All	All
	HS distance	0	0.29	0.12	0.16	0.02

2.3 Existing Readout Calibration Techniques

Table 1 lists the recent studies for readout calibration. They propose different formulations to simplify the noise matrix

characterization, which can be categorized into two types: qubit-independent method and sparsity-aware method. The qubit-independent method includes IBU [50] and Yang et al. [61], which calculates the noise matrix by computing the tensor product of a series of meta-matrices (2×2 matrix of single-qubit). M3 [50] and Nation et al. [38] are sparsity-aware methods that exploit the underlying sparse patterns of the noise matrix. We provide a comparison with regard to scalability and accuracy in Table 1. The accuracy is defined as the Hilbert-Schmidt distance [58] between the real noise matrix M and the formulated matrix M':

$$D_{HS}(M, M') = 1 - \frac{|Tr(M^{\dagger} M')|}{d^2}, \tag{5}$$

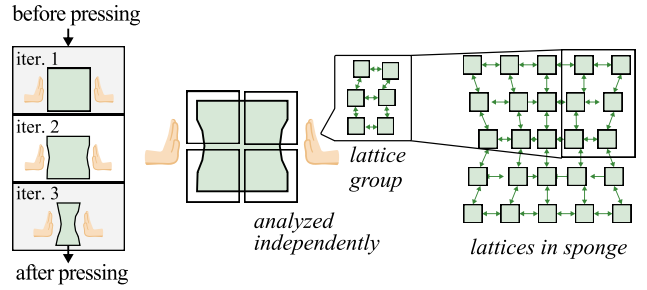
where d is the dimension of the matrices.

In terms of scalability, the qubit-independent methods exhibit a linear number of circuits to characterize the noise matrix, which only needs to execute $2N_q$ benchmarking circuits [50, 61]. However, they have to introduce linear equation solvers [2, 48] to calculate the inverse matrix (calibration matrix), which limits the upper bound to 80 qubits when calibrating without exploiting sparsity. On the other hand, the sparsity-aware methods prune the matrix under a threshold of Hamming distance [37, 38]. However, they show a low compression ratio as most small matrix elements are important for calibration accuracy and cannot be pruned. For example, M3 [37] fails to handle 45-qubit calibration when setting the threshold of Hamming distance to 3, as it requires 1.9PB ($\sim (1000\times\sum_{i=0}^{3} C_{45}^{i})^2\times$8Bit) memory to store the noise matrix in sparse column format. Moreover, the matrices used in M3 need to be characterized for each calibration, which further increases the complexity.

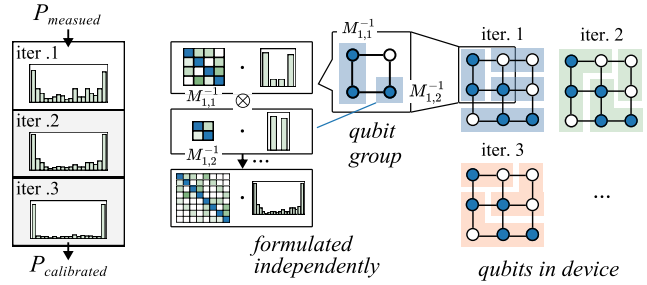
In terms of accuracy, the qubit-independent methods fail to capture the error caused by qubit interactions, e.g., crosstalk. More importantly, the inaccuracy accumulates with the length of the tensor product increases. For example, the scalability of IBU [50] is limited by its low accuracy. As it employs qubit-independent noise matrices, it shows 0.55 Hilbert-Schmidt distance compared to the real noise matrix with 80 qubits on [39], leading to no fidelity improvement in the calibration. Besides, IBU and Yang et al. [61] mainly work on GHZ [20] and BV [7] algorithms because they consist of fewer bit-strings with non-zero probabilities, which limits the generality. The sparsity-aware methods show a smaller Hilbert-Schmidt distance. However, it is still not precise enough for calibration as many important small matrix elements are pruned. For example, it still exhibits around 0.12 Hilbert-Schmidt distance compared to the 80-qubit real noise matrix.

3 QuFEM Formulation

Motivated by the limitations of prior methods [37, 50, 50, 61], we propose QuFEM that is inspired by FEM. Figure 2 (a) shows a typical application of FEM in fluid mechanics that analyzes the deformation of a sponge under compression.



(a) Modeling the deformation of a sponge under compression by FEM.



(b) Methodology of QuFEM inspired from FEM.

Figure 2. Extending traditional FEM to QuFEM.

The force distribution of the sponge exhibits locality. In this case, FEM applies a divide-and-conquer strategy. To simplify the modeling complexity, FEM partitions the sponge into multiple lattices and analyzes each lattice independently. By iteratively putting states of lattices all together, the evolution of the sponge is analyzed.

Inspired by it, QuFEM iteratively calibrates the readout following the FEM workflow. Figure 2 (b) presents an example of it. As the noise between qubits also exhibits locality, we divide qubits into multiple qubit groups. The noise of each group is formulated independently. Moreover, QuFEM adopts the mesh adaption method [5, 19], which employs different grouping schemes in the iterations to thoroughly cover the qubit interactions. In Section 3.1, we present the mathematical equation to calibrate the given measured probability distribution of each circuit. We then introduce the characterization flow and the specific calibration flow to obtain calibration data and compute the equation, respectively.

3.1 Calibration Formulation

QuFEM divides the qubits involved in the measured circuit output into multiple groups. We define the circuit output $P_{measured}$ as P_1, and the calibration output $P_{calibrated}$ as P_{L+1}. QuFEM reformulates the calibration as an iterative process with a series of sub-noise matrices.

$$\begin{aligned} &\text{Iter. 1: } P_2 = (M_{1,1} \otimes M_{1,2} \otimes \cdots M_{1,K})^{-1} P_1, \\ &\text{Iter. 2: } P_3 = (M_{2,1} \otimes M_{2,2} \otimes \cdots M_{2,K})^{-1} P_2, \\ &\qquad \cdots, \\ &\text{Iter. } L\text{: } P_{L+1} = (M_{L,1} \otimes M_{L,2} \otimes \cdots M_{L,K})^{-1} P_L, \end{aligned} \tag{6}$$

where $M_{i,j}$ is the sub-noise matrix of the j^{th} qubit group in the i^{th} iteration. $\otimes$ is the tensor-product to composite these sub-noise matrices. P_2 to P_L serve as the intermediate probability distributions that iteratively approximate the ideal probability distribution. We use $g_{i,j}$ to notate the j^{th} qubit group in the i^{th} iteration. $G_i = \{g_{i,1}, \cdots, g_{i,K}\}$ denotes the grouping scheme of i^{th} iteration. L and K represent the number of the iterations and the number of the groups, respectively.

The i^{th} iteration calibrates P_i to P_{i+1}. We define that $p_i(x)$ is the probability of observing bit-string x in distribution P_i. We have:

$$P_i = \sum p_i(x) \left|x\right\rangle,$$

where each bit-string x can be segmented to sub-bit-strings according to the grouping scheme of the i^{th} iteration :

$$\left|x\right\rangle = \left|x_{i,1}\right\rangle \left|x_{i,2}\right\rangle \cdots \left|x_{i,K}\right\rangle.$$

$x_{i,j}$ is the sub-bit-string of qubit group $g_{i,j}$. We can rewrite the calibration in each iteration according to the segment as follows:

$$\begin{aligned} P_{i+1} &= (M_{i,1} \otimes \cdots \otimes M_{i,K})^{-1} P_i \\ &= (M_{i,1}^{-1} \otimes \cdots \otimes M_{i,K}^{-1}) P_i \\ &= (M_{i,1}^{-1} \otimes \cdots \otimes M_{i,K}^{-1}) \Big(\sum\nolimits_{x \in NZ_i} p_i(x) \left|x_{i,1}\right\rangle \cdots \left|x_{i,K}\right\rangle \Big) \\ &= \sum\nolimits_{x \in NZ_i} p_i(x) \left(M_{i,1}^{-1} \left|x_{i,1}\right\rangle \otimes \cdots \otimes M_{i,K}^{-1} \left|x_{i,K}\right\rangle \right), \end{aligned} \tag{7}$$

where NZ_i is the bit-strings with non-zero probabilities of distribution P_i. For example, assuming that distribution P_1 has two non-zero probabilities of $p_1(000)$ and $p_1(011)$ and the qubits groups include $g_{1,1} = \{q_1, q_2\}, g_{1,2} = \{q_3\}$, Equation (7) is written as:

$$\begin{aligned} P_2 &= (M_{1,1} \otimes M_{1,2})^{-1} P_1 \\ &= p_1(000)(M_{1,1}^{-1} \left|00\right\rangle \otimes M_{1,2}^{-1} \left|0\right\rangle) \\ &+ p_1(011)(M_{1,1}^{-1} \left|01\right\rangle \otimes M_{1,2}^{-1} \left|1\right\rangle), \end{aligned}$$

Equation (7) provides two advantages. First, its time complexity is linear to the size of non-zero probabilities NZ_i, which is typically below the number of shots in the readout. Second, the MVM of each group is independently performed with a constant time as the size of sub-noise matrices is determined.

3.2 QuFEM Overview

QuFEM consists of a characterization flow and a calibration flow, presented in Algorithm 1 and Algorithm 2, respectively. The characterization flow aims to generate the necessary parameters for calibration, i.e., the grouping scheme G_i and the benchmarking probability distributions BP_i in each iteration for sub-noise matrix generation. In line 1, the characterization flow executes a set of benchmarking circuits. The outputs of these circuits serve as the benchmarking probability distributions BP_1 that are used to analyze the noisy perturbation. In lines 2-13, the algorithm then iteratively

Algorithm 1 Characterization flow

Input: quantum device
Output: calibration parameters $CP = [G_1, \cdots, G_L]$, $[BP_1, \cdots, BP_L]$
1: obtain initial benchmarking probability distributions BP_1 by executing benchmarking circuits on the device
2: **for** $i = 1$ to L **do**
3: obtain grouping scheme G_i by partitioning a weighted qubit graph based on BP_i
4: record G_i and BP_i in CP
5: **for** P in BP_i **do**
6: $\{Q_M\}$ = measured qubits of P
7: obtain noise matrix according to $\{Q_M\}$, G_i and BP_i
8: update P based on Equation (7)
9: put P into BP_{i+1}
10: **end for**
11: **end for**

Algorithm 2 Calibration flow

Input: measured probability distribution P_1, calibration parameters $CP = [G_1, \cdots, G_L]$, $[BP_1, \cdots, BP_L]$
Output: calibrated probability distribution P_{L+1}
1: Q_M = measured qubits of P
2: **for** $i = 1$ to L **do**
3: obtain noise matrix based on Q_M, G_i and BP_i
4: calculate P_{i+1} based on Equation 7
5: **end for**

constructs the grouping scheme and the benchmarking probability distributions BP_i for each iteration. Specifically, for i^{th} iteration, in line 1, it constructs a weighted graph and partitions qubits based on BP_i, generating a grouping scheme G_i. G_i and BP_i will be used in the i^{th} iteration in the calibration flow. In lines 6-12, for each probability in BP_i, the algorithm collects the measured qubits and updates the probability distribution using Equation (7).

Algorithm 2 presents the calibration flow, taking the static output from the characterization flow. In line 1, it obtains measured qubits. Starting from P_1, in lines 2-5, it follows Equation (6) to iteratively calibrate P_i to P_{i+1}. In line 2, the calibration obtains the sub-noise matrices based on Q_M and calibration parameters, i.e., G_i and BP_i. Line 3 calculates P_{i+1} based on Equation 7.

For a target quantum device, the calibration parameters are static. Besides, we highlight two features of QuFEM: *1)* the benchmarking probability distributions BP_1 are probability distributions obtained by running benchmarking circuits, while the rest distributions BP_i in each iteration are updated using Equation (7); *2)* the sub-noise matrices are dynamically generated in both characterization flow and calibration flow, according to the measured qubits of the readout to maximize the fidelity, which is motivated by the fact that interactions

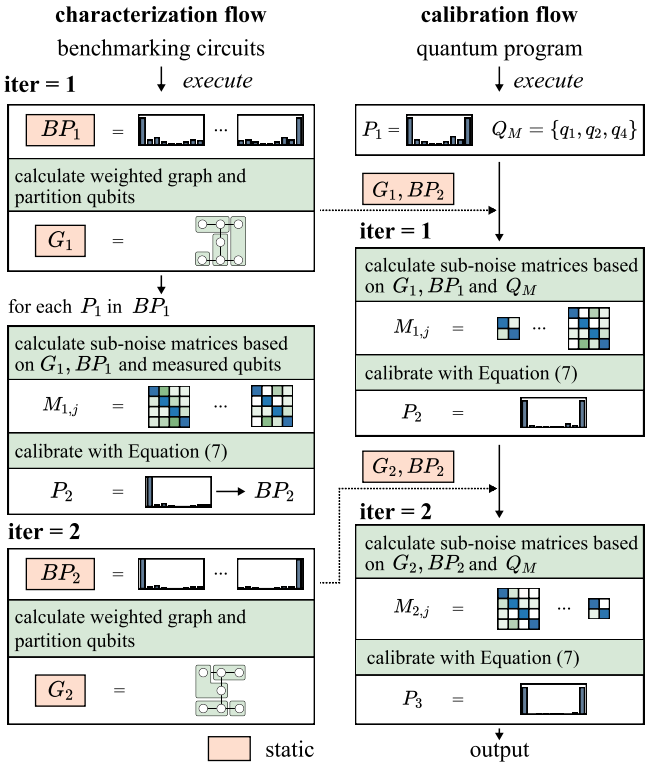


Figure 3. Example of QuFEM with 2 iterations.

always change under different combinations of measured qubits.

Figure 3 presents a 2-iteration example. In the characterization flow, the first iteration obtains a grouping scheme G_1 based on benchmarking probability distribution BP_1. BP_1 and G_1 are stored for the calibration flow. Then, each probability distribution of BP_1 is updated to construct BP_2. The second iteration outputs the partition scheme G_2 of qubits, derived from BP_2. The characterization flow finally outputs $CP = [G_1, G_2], [BP_1, BP_2]$, which serves as calibration parameters. In the calibration flow, given any measured probability distribution P_1 with the measured qubits Q_M, the first iteration obtains the sub-noise matrices based on Q_M, G_1, BP_1, and updates P_1 to P_2 using Equation (7). The second iteration follows the same steps that generate sub-noise matrices based on Q_M, G_2, BP_3, and calibrate P_2 to P_3. P_3 is the resultant probability for this two-iteration calibration.

3.3 Characterization Flow

The characterization flow involves two key techniques. The first technique constructs the weighted graph to quantify qubit interactions and partition qubits, generating the grouping scheme in each iteration. The second technique calculates the sub-noise matrices for calibration.

Weighted graph construction and qubit partition. We apply different grouping schemes in the calibration formulation to cover local interactions between qubits. The schemes

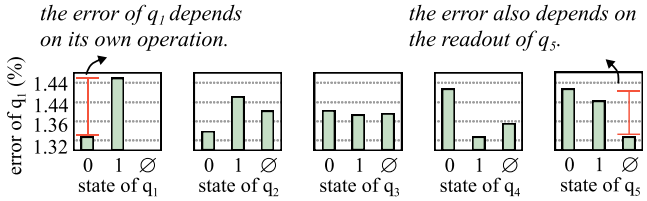


Figure 4. Example of state-dependent and readout-dependent noises on the IBMQ Perth quantum device.

aim to maximize the locality in groups. To this end, we quantify the interactions between qubits by a weighted graph. The quantification is based on the benchmarking probability distributions, initially obtained by executing benchmarking circuits with the methods that will be introduced in Section 4.1. Each qubit is prepared into a state, while sub-sets of them are measured in the circuits. For each shot of the circuit execution, the operation and the readout output of a qubit are recorded as a triple:

- $ideal \in \{0, 1, \varnothing\}$. This value records the ideal (initial) basis state that the qubit is prepared in the circuit.
- $measured \in \{0, 1, \varnothing\}$. This value records the measured output of the qubit in this shot.
- $ef \in \{0, 1\}$. This is the abbreviation of the error flag, where 0 means the measured output matches the ideal result, and 1 means the error occurs.

$\varnothing$ means the qubit is not measured and has no output. In this case, it cannot have readout error ($ideal = \varnothing$, $measured = \varnothing$, $ef = 0$). For example, a benchmarking circuit prepares qubit q_2 into $|1\rangle$, respectively. When the q_1 is not measured and the output of q_2 is $|0\rangle$, the assignments of the triples are:

$$(q_1.ideal = \varnothing,\ q_1.measured = \varnothing,\ q_1.ef = 0),$$
$$(q_2.ideal = 1,\ q_2.measured = 0,\ q_2.ef = 1).$$

According to the recorded triples, we observe the readout error is state-independent, which means that the error varies when the qubit stays in different states. Besides, the interactions from one qubit to another vary under different operations, i.e., state preparation and readout, leading to different readout errors. For example, Figure 4 presents the readout error of qubit q_1 under the prepared states and readout operation of different qubits on the IBMQ Perth quantum device. The readout error of qubit q_1 increases by 0.12% when it is in $|1\rangle$ compared to when it is in $|0\rangle$. The error of qubit q_1 also decreases by around 0.11% when qubit q_5 is not measured. This observation matches the mathematical equation of the readout in Equation 1, where the noisy readout is described as an equation involving both the states and the readout frequency of other qubits.

We quantify the interaction from an operation of a qubit to an operation of another qubit based on this observation.

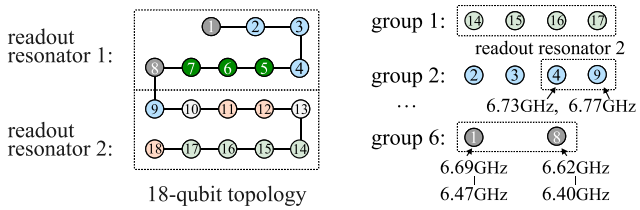


Figure 5. Example of a grouping scheme on our 18-qubit custom quantum device.

An interaction $q_i.ideal = x \longrightarrow q_j.ideal = y$ is defined as:

$$\begin{aligned} &interact(q_i.ideal = x \longrightarrow q_j.ideal = y) = \\ &\left|p(q_j.ef = 1 \mid C1, C2) - p(q_j.ef = 1 \mid C2)\right| \\ &C1:\ q_i.ideal = x,\ C2:\ q_j.ideal = y, \end{aligned} \tag{8}$$

where $x \in \{0, 1, \varnothing\}$, $y \in \{0, 1\}$. Equation (8) characterizes the correlation between the $q_i.ideal = x$ and the readout error of q_j. For example, the interaction

$$interact(q_1.ideal = 0 \longrightarrow q_2.ideal = 1) = \left|2.27\% - 2.08\%\right| = 0.19\%$$

means that qubit q_1 increases the readout error of qubit q_2 by 0.19% (from 2.08% to 2.27%), indicating a potential interaction when qubits q_1 and q_2 are both prepared into state $|1\rangle$. In this manner, we define the weight between two qubits by collecting all cases of interactions.

$$\begin{aligned} weight(q_i, q_j) = &\sum_{x\in\{0,1,\varnothing\}, y\in\{0,1\}} interact(q_i.ideal = x \longrightarrow q_j.ideal = y) \\ &+ \sum_{x\in\{0,1,\varnothing\}, y\in\{0,1\}} interact(q_j.ideal = x \longrightarrow q_i.ideal = y) \end{aligned} \tag{9}$$

We then construct a qubit graph. The edge of the graph is labeled by the weight in Equation (9). We leverage the MAX-CUT solver to partition qubits in the graph into groups [18]. The partition aims to maximize the locality in the group, i.e., the summation of the weights in the group. Such a grouping scheme ensures that, in each iteration, the formulated sub-noise matrices try to comprehensively capture the interactions between qubits, rendering them closer to the real noise matrix.

Figure 5 presents an example of the grouping scheme on the topology of our 18-qubit custom superconducting device [62]. Three cases are observed to make qubits in the same group: qubits that share the same readout resonator (e.g., qubits $q_{14} - q_{17}$); qubits that show similar readout frequencies (e.g., qubits q_4 and q_9); qubits that have overlapping in the shift region of frequencies (e.g., qubits q_1 and q_8). These observations are also demonstrated in the results from other quantum devices and can be used as prior knowledge to facilitate the partition.

Dynamic noise matrix generation. The grouping schemes in different iterations aim to capture different noises. To achieve this, after qubit partition in each iteration, we apply calibration to the benchmarking probability distributions. This eliminates the captured noises in the grouping scheme, so the grouping scheme of the following iterations will focus on the remaining noises.

This calibration is based on group scheme $G_i = \{g_{i,1}, \cdots, g_{i,K}\}$. As interactions vary in the circuits with different measured qubits, we dynamically generate sub-noise matrices for each probability distribution. We define the set of qubits that is measured as Q_M. Then, for each group $g_{i,j}$, we define:

$$g_\cap = Q_M \cap g_{i,j},\ g_\varnothing = g_{i,j} - g_\cap \tag{10}$$

where $g_\cap$ represents the overlapped qubits of $g_{i,j}$ and Q_M. $g_\varnothing$ consists of the rest of the qubits of the group. Similar to Equation (3), the sub-noise matrix formulates the perturbation caused by noise. Its element is defined as:

$$M_{i,j}[x][y] = P(g_\cap.measured = x \mid g_\cap.ideal = y,\ g_\varnothing.ideal = \varnothing).$$

The conditional probability is estimated based on the benchmarking probability distribution BP_i of this iteration. Here, we additionally introduce condition $g_\varnothing.ideal = \varnothing$ to ensure that the matrix generation considers the unmeasured qubits. Moreover, as the measured outputs of qubits only depend on the operations of qubits, we can formulate the equation into:

$$\begin{aligned} M_{i,j}[x][y] = \prod_{q\in g_\cap} &P(q.measured = x_q \mid g_\cap.ideal = y, \\ &g_\varnothing.ideal = \varnothing), \end{aligned} \tag{11}$$

where x_q is the bit of qubit q in bit-string x.

For example, assuming qubit group $g_{1,1} = \{q_1, q_2, q_3\}$, and measured qubit set $Q_m = \{q_1, q_3, q_4\}$, we can obtain that $g_\cap = \{q_1, q_3\}$, and $g_\varnothing = \{q_2\}$. The corresponding matrix element in (3, 0) is

$$\begin{aligned} M_{1,1}[3][0] = &P(q_1.measured = 1 \mid g_\cap.ideal = 00,\ g_\varnothing.ideal = \varnothing) \\ &\cdot P(q_3.measured = 1 \mid g_\cap.ideal = 00,\ g_\varnothing.ideal = \varnothing). \end{aligned}$$

3.4 Calibration Flow

The calibration flow follows the two-step iteration to calibrate the target probability distribution. In each iteration, it first calculates the sub-noise matrices based on Equation (10) and Equation (11). The grouping scheme $g_{i,j}$ of Equation (10) comes from the characterization flow. The conditional probabilities of Equation (11) are computed based on the benchmarking probability distributions BP_i from the characterization flow. Second, the calibration flow applies MVM following Equation (7). If there are remaining iterations, it goes to the next iteration with the MVM result.

The static grouping schemes and calibration datasets are based on the fact that the qubit interactions are constant. This consistency holds in current quantum hardware since current quantum operations (including readout) rely on the ability of the hardware to implement stable (constant) system Hamiltonian. Physically, the interactions are determined after the deployment and do not change unless the readout frequencies of qubits change. For example, the constant interaction of trapped-ion hardware is demonstrated in [42].

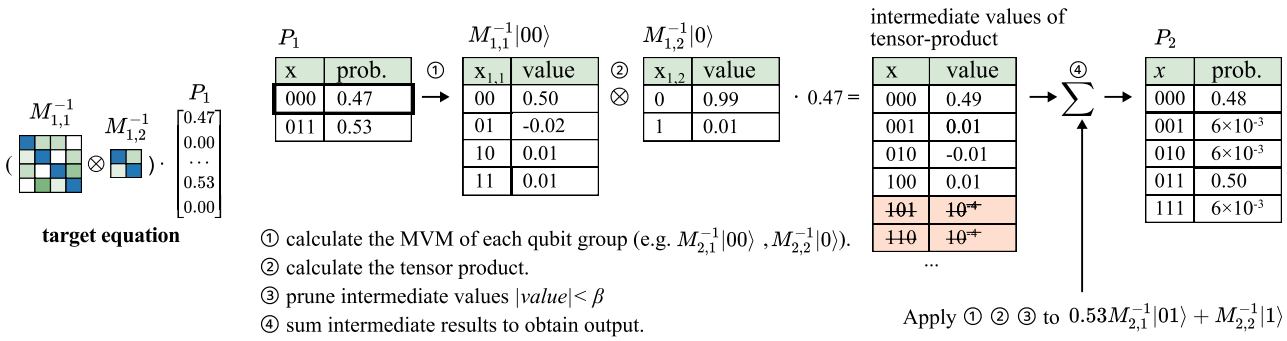


Figure 6. Example of pruning the intermediate values of tensor-product.

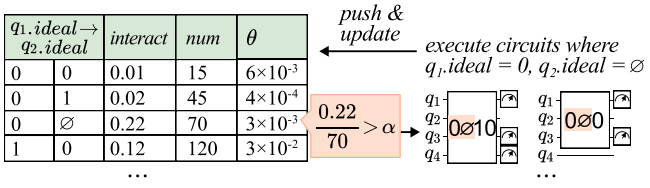


Figure 7. Example of benchmarking circuit generation.

Grouping aims to capture such inherent interactions. Overall, the noise matrix describes the perturbation from the ideal probability distribution to the noisy distribution according to Equation (3). We can follow FEM to model this perturbation inside the qubit groups by small sub-noise matrices. We then adopt different grouping schemes to capture the interactions across the groups.

4 QuFEM Implementation

4.1 Benchmarking Circuit Generation

QuFEM employs more flexible benchmarking circuits to characterize the readout noise accurately. The circuit involves all qubits of the device, where each qubit is operated with three possible options:

1) the qubit is set to $|0\rangle$ and measured;
2) the qubit is set to $|1\rangle$ and measured;
3) the qubit is set to a random state ($|0\rangle$ or $|1\rangle$) and not measured.

Compared to prior works [9, 37, 50] that only have options *1)* and *2)*, QuFEM includes option *3)* to characterize the readout-dependent interactions.

To reduce the number of benchmarking circuits and maintain high accuracy, we identify the circuits that enable the high qubit interactions and profile their execution results, while the circuits with low interactions mainly involve qubit-independent errors so that their outputs can be easily modeled without program executions. Clearly, we start by randomly generating a number of circuits (4 times the number of qubits) to obtain an initial value of interactions between every two qubits. After that, we maintain a table to record the updated interaction values and the numbers of circuits involved in these interactions. For example, in Figure (7), there are 70 benchmarking circuits that apply options *1)* and *3)* to qubits q_1 and q_2, respectively (involving interaction $q_1.ideal = 0 \rightarrow q_2.ideal = \varnothing$). We record $num = 70$ and $interact = 0.22$, where $interact$ is estimated based on these 70 circuits and Equation (8)). Next, we define a metric θ to distinguish the critical interaction.

$$\theta = \frac{interact(q_i.ideal = x \rightarrow q_j.ideal = y)}{num(q_i.ideal = x \rightarrow q_j.ideal = y)} \tag{12}$$

Essentially, the interaction is static and only depends on the physical device. We can estimate it more precisely by executing more benchmarking circuits. In other words, θ naturally decreases with the number of benchmarking circuits. Based on this foundation, we set a threshold α to specify the desired accuracy. For the interaction with $\theta > \alpha$, we iteratively execute circuits involving it and update the table. For example, as shown in Figure 7, there is an interaction that exceeds the threshold. Thus, we prepare a number of circuits with $q_1.ideal = 0$, $q_2.ideal = \varnothing$ and collect their outputs. In this manner, we keep executing circuits until θs of all interactions are less than threshold α. Overall, Equation (12) ensures that to exceed the threshold, significant interactions require more benchmarking circuit executions, while for interactions that are close to 0, the circuit executions are unnecessary as the output distribution of each qubit is the same as the single-qubit benchmarking circuit.

4.2 Sparse Tensor-Product Engine

The iterative calibration based on Equation (7) involves multiple tensor-products of MVM results that have exponential complexity. However, we observe that the non-zero elements of the sub-noise matrix mainly distribute near the diagonal. Such sparsity will accumulate to a higher degree during the tensor-products. We, therefore, propose a sparse tensor-product engine to accelerate the computation. The engine first calculates every MVM of $M_{i,j}^{-1}|x_{i,j}\rangle$. When applying tensor-products to these MVM results, the engine prunes intermediate values based on a threshold β. Figure 6 presents an example. ① For the non-zero probability (0.47) under state

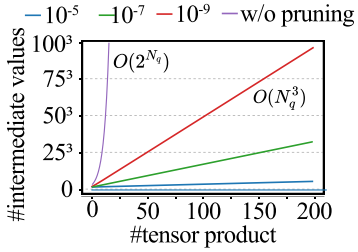


Figure 8. Number of intermediate values exceeding the threshold.

000, the engine calculates the MVM of $M_{1,1}^{-1}|00\rangle$ and $M_{1,2}^{-1}|0\rangle$; ② It then calculates the tensor product of the MVM results; ③ The intermediate values from tensor-product are pruned through a threshold β. Similarly, the engine executes ①②③ for the non-zero probability (0.53) under state 011 and aggregates the pruned intermediate values.

Instead of directly pruning on the noise matrix, the QuFEM engine chooses to prune the intermediate values so that we can achieve higher characterization accuracy. Moreover, the number of pruned intermediate values grows along the chain of multiple tensor-products, which actually brings a huge reduction in computational complexity. To demonstrate this, Figure 8 provides the number of elements exceeding different thresholds along the chain of multiple tensor products, using the real data from benchmarking circuits. We observe that the complexity of our technique lies within a polynomial complexity $\mathcal{O}(N_q^3)$ when the number of tensor products is less than 200. The setting of the threshold is mainly determined by the probability of one bit-flip error in the readout, which usually ranges from 1.0% to 10.0% [14, 43, 52]. Since most previous works model less than 2 bit-flips ($1.0\%^2 = 10^{-4}$), a threshold of 10^{-5} is enough to achieve a higher accuracy.

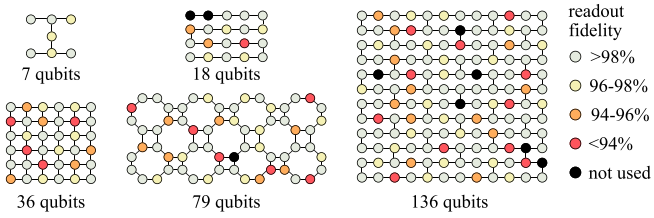
5 QuFEM Complexity

Three parameters of the quantum device determine the complexity of QuFEM: the number of qubits N_q, the readout error rate (quantified via the average interaction $\bar{interact}$), and the number of iterations L. Empirically, we find that setting the maximum number of qubits in the qubit groups to 3 is enough to formulate the noise matrix precisely. Thus, we regard this parameter as a constant value.

The time of the characterization depends on the number of executed circuits, which is determined by the threshold α and the average interaction $\bar{interact}$. There are $6N_q^2$ possible interactions between qubits in total. The worst case is that the benchmarking circuits for these interactions have no overlapping. To reach threshold α, each interaction requires $\mathcal{O}(\alpha/\bar{interact})$ benchmarking circuits. Therefore, the upper bound of the time complexity is $\mathcal{O}(6N_q^2\alpha/\bar{interact})$.

The time and memory consumption of the calibration is concentrated in the matrix generation and the tensor product engine. The conditional probability of the matrix generation has been offline calculated. Since the size of the sub-noise matrix is constant, the matrix generation shows a linear complexity $\mathcal{O}(N_q)$. In MVM, the size of intermediate values is cubic to the number of qubits $\mathcal{O}(N_q^3)$. Overall, for L iterations, the time complexity of the calibration is $\mathcal{O}(LN_q + LN_q^3)$. The memory is mainly used to store sub-noise matrices and intermediate values. As the memory of each iteration can be reused, the overall memory complexity is $\mathcal{O}(N_q + N_q^3)$.

Table 2. Specification of 5 quantum devices used in the evaluation.



Platform	#Qubits	1-q fidelity	2-q fidelity	Instructions
Quafu [39]	136	94.6 ± 3.1%	94.6 ± 3.0%	*ID, RX, RY, RZ, H, CX*
	18	95.9 ± 1.3%	95.9 ± 1.3%	*ID, RX, RY, RZ, H, CX*
Rigetti [46]	79	99.5 ± 1.1%	90.0 ± 6.4%	*CPHASE, XY*
Self-developed	36	99.9 ± 0.1%	98.7 ± 0.8%	*U3, CZ*
IBMQ [44]	7	99.9 ± 0.1%	99.2 ± 0.1%	*CX, ID, RZ, SX, X*

6 Evaluation

6.1 Experimental Setup

Platforms. We evaluate QuFEM on 5 quantum computers (cf. Table 2) with the number of qubits ranging from 7 to 136, where, in particular, the 36-qubit platform is based on our self-developed device consisting of 36 Xmon qubits arranged in a 6×6 grid topology. We execute the benchmarking circuits on these platforms and then characterize the noise matrix. All software experiments are performed on a server with two AMD EPYC 2.25GHz 64-core CPUs and 1.6TB memory. All programs use a single thread on these CPUs. The memory usage is profiled by the `memory_profiler` package [16]. The time- and memory-out are set to 10 hours and 1.0TB, respectively; we provide rough estimations (marked by ~) of the required time, memory, and number of benchmarking circuits in case of time/memory-out.

QuFEM implementation and configuration. We have implemented QuFEM with `Python` (3.9.13) and the `NumPy` package (1.23.1). In the default configuration of QuFEM, we set both the iteration number L and the maximum number of qubits K in a group to 2. The thresholds are set as $\alpha = 2.5\times10^{-5}$ for matrix chracterizaion and $\beta = 10^{-5}$ for pruning. Alternative parameter settings are evaluated in Section 6.4. Each circuit is sampled 2000 times to obtain the probability distribution.

Baselines. We compare QuFEM against state-of-the-art readout calibration approaches, including IBU [50], M3 [37],

Table 3. Number of circuits used for readout characterization.

#Qubits	IBU [50]	CTMP [9]	M3 [37]	Golden (baseline)	QuFEM
7	14	14	128	128	110
18	36	36	8356	$\sim 2.5 \times 10^5$	594
27	54	54	73813	$\sim 5.8 \times 10^{10}$	704
36	~ 72	~ 72	131064	$\sim 6.8 \times 10^{17}$	720
49	~ 98	~ 98	$\sim 2.4 \times 10^5$	$\sim 5.8 \times 10^{14}$	728
79	~ 158	~ 158	$\sim 6.3 \times 10^5$	$\sim 6.0 \times 10^{24}$	826
136	~ 272	~ 272	$\sim 1.8 \times 10^6$	$\sim 8.7 \times 10^{41}$	1380
N_q	$\mathcal{O}(2N_q)$	$\mathcal{O}(2N_q)$	$\mathcal{O}(N_q^{3.1})$	$\mathcal{O}(2^{N_q})$	$\mathcal{O}(7.6N_q)$

CTMP [9], and Q-BEEP [53]. The baseline is defined as the golden (real) noise matrix derived from Equation (3). For M3 and IBU, we set the threshold of Hamming distances to 3. We set the iteration number of IBU to 10^5 and its tolerance threshold to 10^{-5}, respectively. For Q-BEEP, the number of iterations1 is set to 20.

Benchmarks. We evaluate QuFEM over 7 well-known quantum algorithms, including GHZ [20], variational quantum classifier (VQC) [17], Bernstein-Vazirani (BV) [7], Simon's [15], quantum support vector machines (QSVM) [51], Hamiltonian simulation (HS), and Deutsch-Jozsa (DJ) algorithms. These algorithms are executed only on the 7-qubit and the 18-qubit platforms, where the aim is to quantify the fidelity improvement after calibration. The real fidelity is defined as the Hellinger fidelity [30] capturing its similarity to the noise-free simulated distribution. Considering that the fidelity turns neglectable on circuits with a large number of qubits, for platforms with more than 18 qubits, we evaluate the calibration time and memory usage of these methods through 1000 probability distributions in the shape of Gaussian (30%), uniform (30%), and spike-like (40%) distributions; each distribution involves 200 bit-strings with non-zero probability.

6.2 Evaluation of Scalability

Characterization step. Table 3 depicts the number of benchmarking circuits executed for the readout characterization. To better present the trend of the computation time as the number of qubits changes, we apply interpolation to the results based on the evaluated quantum devices. The characterization of M3 is conducted for each algorithm output, and therefore we report the average number of benchmarking circuits for calibrating the algorithm outputs. Compared to M3 which has the time complexity of $\mathcal{O}(N_q^{3.1})$, QuFEM exhibits a lower (linear) time complexity of $\mathcal{O}(7.6N_q)$, thereby yielding, e.g., on the 136-qubit device, a reduction in the number of circuits from 1.8×10^6 to 1380 (1, 304.3× reduction). These 1380 circuits can be executed in 13.8 seconds. Such reduction is attributed to the technique in QuFEM for eliminating redundant circuit executions. Observe that the qubit-independent methods IBU and CTMP require only 272 circuits on the 136-qubit device, but they inherently ignore the crosstalk and thus lead to potential calibration failures (see Section 6.3). QuFEM requires around 5× benchmarking circuits compared to IBU and CTMP. However, it provides the ability to model the qubit interaction and thereby achieve higher fidelity.

Table 4. Calibration time (in seconds) on the classical computer.

#Qubits	IBU [50]	CTMP [9]	M3 [37]	Q-BEEP [53]	QuFEM
7	0.37	0.017	0.024	0.029	0.029
18	1.52	30.56	1.78	92280.92	**0.70**
27	6.45	5231.45	134.14	$\sim 2.6 \times 10^7$	**1.48**
36	101.56	$\sim 1.9 \times 10^6$	645.11	$\sim 7.6 \times 10^9$	**2.20**
49	509.47	$\sim 3.7 \times 10^8$	$\sim 5.4 \times 10^3$	$\sim 2.8 \times 10^{13}$	**5.56**
79	1924.27	$\sim 6.5 \times 10^{13}$	$\sim 1.1 \times 10^6$	$\sim 4.8 \times 10^{21}$	**24.58**
136	$\sim 4.2 \times 10^5$	$\sim 6.0 \times 10^{23}$	$\sim 2.3 \times 10^{10}$	$\sim 2.1 \times 10^{37}$	**169.65**
N_q	$\mathcal{O}(1.1^{N_q})$	$\mathcal{O}(1.5^{N_q})$	$\mathcal{O}(1.2^{N_q})$	$\mathcal{O}(1.8^{N_q})$	$\mathbf{\mathcal{O}(N_q^2)}$

Table 5. Memory consumption in megabytes.

#Qubits	IBU [50]	CTMP [9]	M3 [37]	Q-BEEP [53]	QuFEM
7	73.27	0.19	34.10	38.80	5.10
18	78.84	45.56	17953.20	193.61	**8.40**
27	84.54	271.24	108751.20	~ 318.98	**16.20**
36	98.30	$\sim 2.3 \times 10^3$	331243.45	~ 444.35	**46.90**
49	124.56	$\sim 3.8 \times 10^4$	$\sim 6.2 \times 10^6$	~ 625.44	**90.41**
79	285.74	$\sim 2.6 \times 10^7$	$\sim 1.6 \times 10^7$	$\sim 1.0 \times 10^3$	**127.14**
136	$\sim 9.7 \times 10^3$	$\sim 6.1 \times 10^{12}$	$\sim 4.7 \times 10^8$	$\sim 1.8 \times 10^3$	**366.42**
N_q	$\mathcal{O}(1.1^{N_q})$	$\mathcal{O}(1.2^{N_q})$	$\mathcal{O}(1.2^{N_q})$	$\mathcal{O}(13.9N_q)$	$\mathcal{O}(N_q^3)$

MVM step. Table 4 depicts the calibration overhead averaged over the 7 algorithms. It is shown that, on all the evaluated configurations, QuFEM exhibits the highest calibration efficiency; in fact, it features an *exponential speedup* in terms of time complexity: For instance, QuFEM calibrates the 18-qubit algorithm outputs in only 0.70 second, yielding a speedup of $1.3\times10^5\times$, 2.5×, 43.7×, and 2.2× against Q-BEEP, M3, CTMP, and IBU, respectively. Such speedup turns more significant as the number of qubits increases, e.g., $5.0 \times 10^{12}\times$ against Q-BEEP on the 49-qubit fabricated distributions. The efficiency of QuFEM essentially benefits from the use of small noise matrices and the pruning strategy, whilst M3, IBU, and CTMP represent the noisy readout process as a single noise matrix. For Q-BEEP, the exponential complexity stems from the procedure of iteratively updating a state graph with an exponentially increasing number of nodes.

Table 5 presents the memory consumption of different approaches, amongst which QuFEM requires the least amount of memory without ever reaching the time limit (cf. Q-BEEP, which requires estimatedly less memory yet times out on 49-, 79-, and 136-qubit devices). For example, the calibration

Table 6. The calibration time of QuFEM (in seconds) on the 200- to 500-qubit quantum computers.

Distribution	200 qubits	300 qubits	400 qubits	500 qubits
Gaussian	30.79	37.76	41.07	46.67
Spike-like	38.79	42.32	48.40	55.46
Uniform	40.29	51.65	64.16	75.44
Average	36.62	43.91	51.21	59.19

via QuFEM on the 18-qubit circuit output requires 8.4MB of memory, amounting to 2,137.3×, 9.4×, and 23.0× memory reduction against M3, IBU, and Q-BEEP, respectively. Similarly, on the 136-qubit circuit output, QuFEM requires 0.4GB of memory, while M3 and IBU are estimated to use around 470.0TB and 9.7GB, respectively. Notice that, during calibration, the memory is mainly used to store noise matrices and probability distributions. To reduce memory usage, QuFEM employs significantly small matrices and applies pruning to reduce the size of the probability distributions. As an example, for 136-qubit calibration via QuFEM, the total size of the noise matrices of each iteration is

$$\underbrace{136}_{\#\text{qubits}} \Big/ \underbrace{2}_{\#\text{qubits per groups}} \times \underbrace{2}_{\#\text{iterations}} \times \underbrace{4\times 4}_{\text{matrix size}} = 2,176.$$

In contrast, the size of the M3 noise matrix is 4.7×10^8 on average since the space of bit strings grows exponentially in the number of qubits with Hamming distance 3.

MVM for future quantum devices. To further demonstrate the scalability, we simulate QuFEM calibration on 200- to 500-qubit fabricated distributions; the corresponding calibration time is shown in Table 6. Due to limited time and memory usage, the aforementioned state-of-the-art methods do not calibrate on future quantum devices. We configure the levels of readout error and crosstalk to be the same as on the 136-qubit real-world device. As reported in Table 6, QuFEM can calibrate the 500-qubit circuit outputs in around 1 minutes. It is also worth noting that, among the three types of distribution, the uniform distribution requires the most computational time, e.g., the calibration of the 500-qubit uniform distribution takes 1.4× and 1.6× longer than the spike-like and the Gaussian distributions, respectively. This may stem from the fact that the spike-like and the Gaussian distributions have more bit-strings with small probabilities, thus generating more small values that can be pruned during the computation.

6.3 Evaluation of Accuracy

Fidelity improvement of various algorithms. In Figure 9 (a), we evaluate the calibration performance of QuFEM in terms of accuracy against the competitors on a 7-qubit device. To this end, we define the *relative fidelity* of a circuit output as

$$\frac{\text{fidelity after calibration}}{\text{fidelity before calibration}}. \tag{13}$$

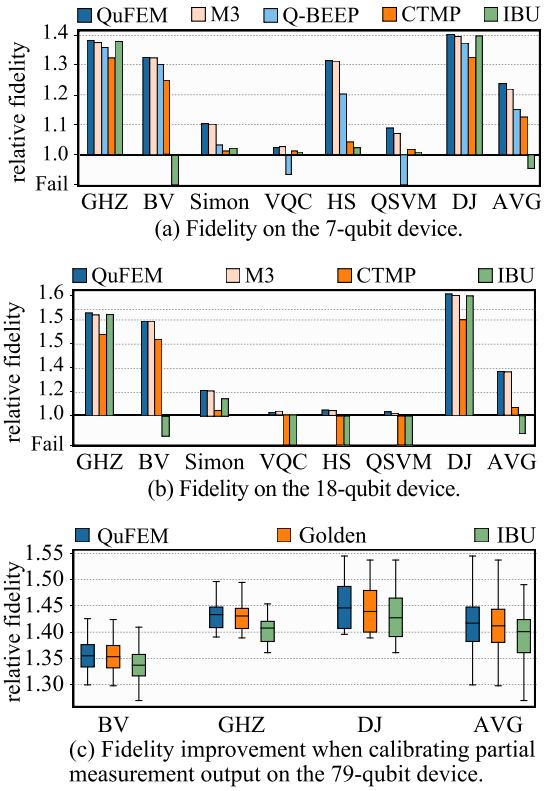


Figure 9. Evaluation on the fidelity improvement.

Overall, QuFEM exhibits an average of 1.004×, 1.1×, 1.1×, and 1.3× relative fidelity improvement compared to M3, Q-BEEP, CTMP, and IBU, respectively. This result suggests that QuFEM can more accurately model the readout evolution.

We carry the comparison further to an 18-qubit device (on which Q-BEEP times out), as shown in Figure 9 (b). In this case, QuFEM also demonstrates an average improvement in relative fidelity of 1.003×, 1.2×, and 1.4× compared to M3, CTMP, and IBU, respectively. Compared to the 7-qubit setting, the fidelity turns worse on the 18-qubit device when we do not calibrate because it has more gates and crosstalk.

Remark that the relative fidelity falling below 1 marks *calibration failure*. In Figure 9 (a), Q-BEEP fails the calibration for VQC and QSVM because it is tailored for specific algorithms like GHZ and BV that show specific structures in the measured distributions; IBU fails the calibration for it does not consider crosstalk noises. The latter turns more prominent in Figure 9 (b) – especially for the VQC and QSVM algorithms, which have a large number of 2-qubit gates inducing even higher readout crosstalk noises – where both IBU and CTMP encounter calibration failures. In contrast, the relative fidelities of QuFEM on the 18-qubit device are stable for VQC and QSVM.

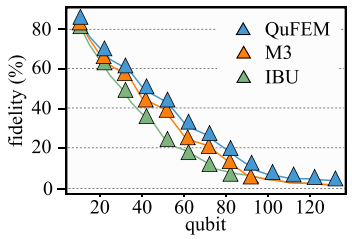


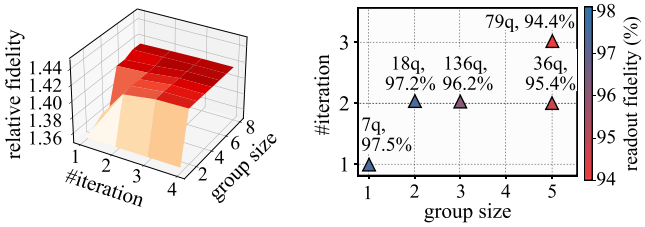
Figure 10. Evaluation in 10-qubit to 131-qubit GHZ outputs.

Fidelity improvement on the partial measurement output. QuFEM has the ability to generate noise matrices for different combinations of qubits, considering the variance of crosstalk. We compare QuFEM to the golden-matrix-based calibration and IBU on the 79-qubit device of Rigetti. To better demonstrate the variance of readout crosstalk, we choose the algorithms that involve higher crosstalk noise, including BV, GHZ, and DJ algorithms. 10 circuits are executed for each algorithm. These circuits measure 10 logical qubits. The mapping from logical qubits to physical qubits is determined through a random selection. As a result, QuFEM shows an average 1.4× fidelity improvement in different levels of crosstalk noise, as shown in Figure 9 (c). This suggests that QuFEM has stable fidelity improvement in different combinations of qubits. In addition, QuFEM (1.43) also has an average 0.03 relative fidelity improvement compared to IBU (1.40). In these algorithms, we even observe that the calibration performance of QuFEM exceeds calibration with the golden noise matrix. This is because QuFEM can better characterize the noise matrix with a limited number of circuit executions, while the conventional method to characterize the golden matrix suffers from the trade-off between accuracy and efficiency.

Evaluation in 10-qubit to 131-qubit GHZ outputs. We compare the fidelity of the GHZ outputs calibrated by QuFEM, M3, and IBU in Figure 10. IBU shows the minimum fidelity as it ignores the qubit interactions, which shows no fidelity improvement when the number of qubits reaches 80, while QuFEM shows the maximum fidelity, achieving 1.6× fidelity improvement compared to M3 (from 11.70% to 18.86%) and 2.8× improvement compared to IBU (from 6.79% to 18.86%).

6.4 Evaluation of Parameter Setting

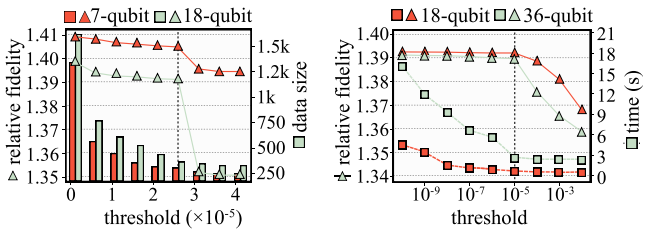
Evaluation of the number of qubits in the group and the number of iterations. Figure 11 (a) evaluates the maximum number of qubits in the group and the number of iterations L when calibrating the algorithm outputs on the 18-qubit device. We observe that setting both parameters to 2 is enough for QuFEM to reach the maximum fidelity improvement on the 18-qubit Quafu quantum device, after which the fidelity improvement converge. This is attributed to our MAX-CUT-based partition strategy that aims to maximize the interaction formulated in the qubit groups.



(a) Fidelities under different settings on the 18-qubit device.

(b) Optimal parameter settings on different quantum devices.

Figure 11. Evaluation of different settings of the number of iterations and the group size.



(a) Characterization threshold α.

(b) Pruning threshold β

Figure 12. Evaluation of different thresholds.

We define the optimal parameters to calibrate a quantum device as the configuration that uses the minimum calibration time to achieve the maximum fidelity Improvement. We present these optimal parameters on different quantum devices in Figure 11 (b). We observe that the configuration mainly depends on the average readout error rate of qubits instead of the number of qubits. For example, the 136-qubit Quafu quantum device has the most qubits, but it requires a smaller number of qubits in the groups (3) compared to the 36-qubit self-developed device (5) as it has a low level of readout noise. This may result from the fact that the qubit interactions show locality in the processor topology.

Evaluation of threshold α in generating the benchmarking circuits. Compared to M3, QuFEM requires fewer benchmarking circuits. Clearly, according to Figure 12 (a), we observe when increasing the threshold, a smaller number of benchmarking circuits are needed to fulfill the stopping conditions, which may lead to less accuracy in the noise matrix generation. When the characterization threshold increases from 10^{-7} to 2.5×10^{-5}, the relative fidelity remains virtually unchanged and the number of required circuits is reduced 4.4× on the 18-qubit device, respectively. There is also a similar trend on the 7-qubit device. For both devices, 2.5×10^{-5} could be a sweet point, after which there is a large fidelity drop.

Evaluation of threshold β in the pruning. Figure 12 (b) tests different pruning thresholds on the 18-qubit and 36-qubit devices. Setting the threshold to 10^{-5} achieves 5.5×

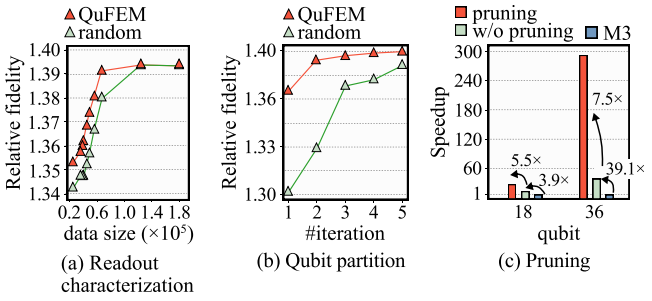


Figure 13. Ablation study of the techniques used in QuFEM, (a) and (b) are performed on the 7-qubit device.

speedup on the 18-qubit device, which leads to a 0.001 fidelity reduction compared to the calibration with golden fidelity. We prune these small elements that originate in noise, leading to non-obvious fidelity reduction. We observe that speedup is significant on quantum devices with more qubits. For example, when the pruning threshold is set to 10^{-5}, the speedup on the 36-qubit device is 1.4× than the 18-qubit device. This is because they have higher proportions of small elements that can be pruned on a 36-qubit quantum computer. Compared to the drop in the computation time, the accuracy does not decrease until the pruning threshold is set to 10^{-5}, which could be a good choice to balance the efficiency and the accuracy.

6.5 Ablation Study

Ablation of the generation of the benchmarking circuits. Figure 13 (a) illustrates the comparison between randomly generating the benchmarking circuits and selecting them by QuFEM. On the 7-qubit device, when both QuFEM and the random selection reach the maximum fidelity, QuFEM shows a 1.7× reduction of data size (713) compared to the random method (1231), which effectively reduces the overall search space. The improvement demonstrates that QuFEM has the ability to identify the benchmarking circuit, capturing a higher degree of qubit interactions.

Ablation of grouping schemes. Figure 13 (b) presents the comparison between the random grouping scheme and the QuFEM grouping scheme. The results show that QuFEM has a 0.05 fidelity improvement (1.36) compared to the random partition strategy (1.31) when the number of iterations is 1. This can be attributed to effectively identifying the great crosstalk in each qubit group. Moreover, QuFEM only uses 2 iterations to reach a close-to-optimal improvement, while the random strategy takes more than 5 iterations.

Ablation of pruning. Figure 13 (c) evaluates the acceleration of our sparse tensor-product engine compared to M3 [37]. On the 18-qubit device, benefiting from the finite element method, QuFEM provides a 3.9× speedup compared to M3. By pruning the small intermediate values below the threshold, QuFEM provides an additional 5.5× speedup. In addition, the pruning results in better speedups on the 36-qubit device, reaching 293.3× overall speedup.

7 Related Work

Matrix-based readout calibration. Current matrix-based techniques mainly focus on improving the scalability [9, 37, 50] and accuracy [8, 36, 50]. To improve the scalability, methods like M3 [37] and Yang. etc. [61] exploit the sparsity of the assignment matrix, while they show limitation as few matrix elements can be pruned. Other methods model only qubit-independent error [9, 50], leading to a large accuracy loss. Bayesian unfolding [8, 36, 50] aims to improve the limited calibration accuracy due to using single circuit output to fill in the matrix elements. QuFEM employs conditional probabilities to estimate fewer benchmarking circuit outputs while providing higher accuracy compared to these methods.

Other readout error optimization techniques. Readout error has been observed to be a dominant source of error that requires careful calibration [40]. The optimization of it can be conducted by exploiting the patterns of the circuit outputs [22, 53, 55], partial measurement [14], and neural network [29, 33] Recently, many meaningful works have been proposed to characterize the noisy process [9, 22, 26, 57]. For example, the relaxation error can be detected by tracing the evolution during the measurement is observed to increase the readout fidelity [33]. Besides, [26] models the error in BV and quantum amplitude estimation algorithms and calibrates outputs by numerical simulation. QuFEM can be applied before these techniques to provide further readout error optimization.

8 Conclusion

Quantum readout error emerges as the predominant source of errors, which greatly reduces the measurement fidelity. The efficacy of matrix-based calibration has been substantiated across diverse quantum platforms. Nonetheless, prevailing methodologies suffer from limitations either in terms of scalability or precision. Drawing inspiration from FEM, we propose QuFEM to calibrate the measurement error. First, we formulate the calibration as a series of tensor-product operations involving sub-noise matrices, where the matrix is iteratively updated with the probability distribution. Then, we introduce how to generate the benchmarking circuits with low time complexity for characterization. Finally, a sparse tensor-product engine exploits the inherent sparsity during the tensor-product to provide an end-to-end acceleration for calibration.

Acknowledgments

This work was supported in part by Zhejiang Pioneer (Jianbing) Project (No. 2023C01036). This work was also funded by the National Key Research and Development Program of China (No. 2023YFF0905200). We would like to thank Song

Chao for sharing superconducting quantum devices and providing insightful advice.

References

[1] Niels Aage, Erik Andreassen, Boyan S Lazarov, and Ole Sigmund. Gigavoxel computational morphogenesis for structural design. *Nature*, 550(7674):84–86, 2017.

[2] E Anderson, Z Bai, C Bischof, J Demmel, J Dongarra, J DuCroz, A Greenbaum, S Hammarling, A McKenney, and D Sorensen. Lapack: A portable line ar al ge br a li br ary fo r hi g h-pe rfor ma n ce co mput ers. 1990.

[3] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, Willliam Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P Harrigan, Michael J Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S Humble, Sergei V Isakov, Evan Jeffrey, Zhan Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandra, Jarrod R McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthhew Neeley, Charles Neil, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C Platt, Chris Quintana, Eleanor G Rieffel, Pedram Roushan, Nicholas C Rubin, Daniel Sank, Kevin J Satzinger, Vadim Smelyanskiy, Kevin J Sung, Matthew D Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, pages 505–510, 2019.

[4] Ramin Ayanzadeh, Narges Alavisamani, Poulami Das, and Moinuddin Qureshi. Frozenqubits: Boosting fidelity of qaoa by skipping hotspot nodes. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'23)*, pages 311–324, 2023.

[5] Timothy J Baker. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements in Analysis and Design*, 25(3-4):243–273, 1997.

[6] Rami Barends, Julian Kelly, Anthony Megrant, Daniel Sank, Evan Jeffrey, Yu Chen, Yi Yin, Ben Chiaro, Josh Mutus, Charles Neill, P. O'Malley, P. Roushan, J. Wenner, T. C. White, A. N. Cleland, and John M. Martinis. Coherent josephson qubit suitable for scalable quantum integrated circuits. *Physical Review Letters*, page 080502, 2013.

[7] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, pages 1510–1523, 1997.

[8] Volker Blobel. Unfolding methods in particle physics. 2011.

[9] Sergey Bravyi, Sarah Sheldon, Abhinav Kandala, David C Mckay, and Jay M Gambetta. Mitigating measurement errors in multiqubit experiments. *Physical Review A*, 103(4):042605, 2021.

[10] A Burrell. *High fidelity readout of trapped ion qubits*. PhD thesis, Oxford University, UK, 2010.

[11] Laura Clinton, Johannes Bausch, and Toby Cubitt. Hamiltonian simulation algorithms for near-term quantum hardware. *Nature communications*, pages 1–10, 2021.

[12] Poulami Das, Eric Kessler, and Yunong Shi. The imitation game: Leveraging copycats for robust native gate selection in nisq programs. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 787–801. IEEE, 2023.

[13] Poulami Das, Swamit Tannu, Siddharth Dangwal, and Moinuddin Qureshi. Adapt: Mitigating idling errors in qubits via adaptive dynamical decoupling. In *Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 950–962, 2021.

[14] Poulami Das, Swamit Tannu, and Moinuddin Qureshi. Jigsaw: Boosting fidelity of nisq programs via measurement subsetting. In *Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 937–949, 2021.

[15] S Dr. On the power of quantum computation. *SIAM Journal on Computing*, page 26, 1997.

[16] Philippe Gervais Fabian Pedregosa. Memory profiler. https://github.com/pythonprofilers/memory_profiler, January 2023.

[17] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[18] Paola Festa, Panos M Pardalos, Mauricio GC Resende, and Celso C Ribeiro. Randomized heuristics for the max-cut problem. *Optimization methods and software*, 17(6):1033–1058, 2002.

[19] Pascal-Jean Frey and Frédéric Alauzet. Anisotropic mesh adaptation for cfd computations. *Computer methods in applied mechanics and engineering*, 194(48-49):5068–5082, 2005.

[20] DM Greenberger, MA Horne, and A Zeilinger. Going beyond Bell's theorem, in "Bell's theorem, quantum theory, and conceptions of the universe", m. kafakos, editor, vol. 37 of. *Fundamental Theories of Physics*, 1989.

[21] Kathleen E Hamilton, Tyler Kharazi, Titus Morris, Alexander J McCaskey, Ryan S Bennink, and Raphael C Pooser. Scalable quantum processor noise characterization. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 430–440. IEEE, 2020.

[22] Rebecca Hicks, Christian W Bauer, and Benjamin Nachman. Readout rebalancing for near-term quantum computers. *Physical Review A*, 103(2):022407, 2021.

[23] IBMQ. Qiskit document: Measurement error mitigation. 2022. https://qiskit.org/documentation/stable/0.26/tutorials/noise/3_measurement_error_mitigation.html.

[24] Mostafa Khezri, Justin Dressel, and Alexander N Korotkov. Qubit measurement error from coupling with a detuned neighbor in circuit qed. *Physical Review A*, 92(5):052306, 2015.

[25] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, page 021318, 2019.

[26] Hyeokjea Kwon and Joonwoo Bae. A hybrid quantum-classical approach to mitigating measurement errors in quantum algorithms. *IEEE Transactions on Computers*, 70(9):1401–1411, 2020.

[27] Thi Ha Kyaw, Tim Menke, Sukin Sim, Abhinav Anand, Nicolas PD Sawaya, William D Oliver, Gian Giacomo Guerreschi, and Alán Aspuru-Guzik. Quantum computer-aided design: digital quantum simulation of quantum processors. *Physical Review Applied*, 16(4):044042, 2021.

[28] David Layden, Guglielmo Mazzola, Ryan V Mishmash, Mario Motta, Pawel Wocjan, Jin-Sung Kim, and Sarah Sheldon. Quantum-enhanced Markov chain Monte Carlo. *Nature*, 619(7969):282–287, 2023.

[29] Benjamin Lienhard, Antti Vepsäläinen, Luke CG Govia, Cole R Hoffer, Jack Y Qiu, Diego Ristè, Matthew Ware, David Kim, Roni Winik, Alexander Melville, Bethany Niedzielski, Jonilyn Yoder, Guilhem J Ribeill, Thomas A Ohki, Hari K Krovi, Terry P Orlando, Simon Gustavsson, and William D Oliver. Deep-neural-network discrimination of multiplexed superconducting-qubit states. *Physical Review Applied*, 17(1):014024, 2022.

[30] Shunlong Luo and Qiang Zhang. Informational distance on quantum-state space. *Physical Review A*, page 032106, 2004.

[31] Filip B Maciejewski, Zoltán Zimborás, and Michał Oszmaniec. Mitigation of readout noise in near-term quantum devices by classical

post-processing based on detector tomography. *Quantum*, 4:257, 2020.

[32] Chris Macklin, K O'brien, D Hover, ME Schwartz, V Bolkhovsky, X Zhang, WD Oliver, and I Siddiqi. A near–quantum-limited josephson traveling-wave parametric amplifier. *Science*, 350(6258):307–310, 2015.

[33] Satvik Maurya, Chaithanya Naik Mude, William D Oliver, Benjamin Lienhard, and Swamit Tannu. Scaling qubit readout with hardware efficient machine learning architectures. In *2023 ACM/IEEE 50th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–13, 2023.

[34] Alex Mott, Joshua Job, Jean-Roch Vlimant, Daniel Lidar, and Maria Spiropulu. Solving a higgs optimization problem with quantum annealing for machine learning. *Nature*, 550(7676):375–379, 2017.

[35] Prakash Murali, David C McKay, Margaret Martonosi, and Ali Javadi-Abhari. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'20)*, pages 1001–1016, 2020.

[36] Benjamin Nachman, Miroslav Urbanek, Wibe A de Jong, and Christian W Bauer. Unfolding quantum computer readout noise. *npj Quantum Information*, 6(1):84, 2020.

[37] Paul D Nation, Hwajung Kang, Neereja Sundaresan, and Jay M Gambetta. Scalable mitigation of measurement errors on quantum computers. *PRX Quantum*, 2(4):040326, 2021.

[38] PD Nation, JR Johansson, MP Blencowe, and AJ Rimberg. Iterative solutions to the steady-state density matrix for optomechanical systems. *Physical Review E*, 91(1):013307, 2015.

[39] Beijing Academy of Quantum Information Sciences. Quafu quantum cloud computing platform. *https://quafu.baqis.ac.cn/*, 2022.

[40] Tirthak Patel and Devesh Tiwari. Disq: a novel quantum output state classification method on ibm quantum computers using openpulse. In *Proceedings of the 39th International Conference on Computer-Aided Design(ICCAD)*, pages 1–9, 2020.

[41] Thomas Picot, A Lupaşcu, S Saito, CJPM Harmans, and JE Mooij. Role of relaxation in the quantum measurement of a superconducting qubit using a nonlinear oscillator. *Physical Review B*, 78(13):132508, 2008.

[42] Ch Piltz, Th Sriarunothai, AF Varón, and Ch Wunderlich. A trapped-ion-based quantum byte with 10- 5 next-neighbour cross-talk. *Nature communications*, 5(1):4679, 2014.

[43] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, page 79, 2018.

[44] IBM Quantum. Ibm quantum systems and simulators. *https://quantum-computing.ibm.com*, 2022.

[45] P. Rebentrost, M. Mohseni, and S. Lloyd. Quantum support vector machine for big feature and big data classification. *Physical Review Letters*, page 130503, 2013.

[46] LLC Rigetti & Co. Rigetti computing: Quantum computing. *https://www.rigetti.com/*, 2022.

[47] Rigetti & Co, LLC. *pyQuil document v2.28.2: QuantumComputer.calibrate*, 2019. https://pyquil-docs.rigetti.com/en/v2.28.2/apidocs/autogen/pyquil.api.QuantumComputer.calibrate.html.

[48] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[49] Tareq Saeed, Ibrahim Abbas, and Marin Marin. A gl model on thermoelastic interaction in a poroelastic material using finite element method. *Symmetry*, 12(3):488, 2020.

[50] KJ Satzinger, Y-J Liu, A Smith, C Knapp, M Newman, C Jones, Z Chen, C Quintana, X Mi, A Dunsworth, C Gidney, I Aleiner, F Arute, K Arya, J Atalaya, R Babbush, JC Bardin, R Barends, J Basso, A Bengtsson, A Bilmes, M Broughton, BB Buckley, DA Buell, B Burkett, N Bushnell, B Chiaro, R Collins, W Courtney, S Demura, AR Derk, D Eppens, C Erickson, L Faoro, E Farhi, AG Fowler, B Foxen, M Giustina, A Greene, JA Gross, MP Harrigan, SD Harrington, J Hilton, S Hong, T Huang, WJ Huggins, LB Ioffe, SV Isakov, E Jeffrey, Z Jiang, D Kafri, K Kechedzhi, T Khattar, S Kim, PV Klimov, AN Korotkov, F Kostritsa, D Landhuis, P Laptev, A Locharla, E Lucero, O Martin, JR Mcclean, M Mcewen, KC Miao, M Mohseni, S Montaeri, W Mruczkiewicz, J Mutus, O Naaman, M Neeley, C Neill, MY Niu, TE Obrien, A Opremcak, B Pato, A Petukhov, NC Rubin, D Sank, V Shvarts, D Strain, M Szalay, C Neill, MY Niu, TE Obrien, A Opremack, B Pato, A Petukhov, NC Rubin, D Sank, V Shvarts, D Strain, M Szalay, B VillaloNnga, TC White, Z Yao, P Yeh, J Yoo, A Zalcman, H Neven, S Boixo, A Megrant, Y Chen, J Kelly, V Smelyanskiy, A Kitaev, M Knap, F Pollmann, and P roushan. Realizing topologically ordered states on a quantum processor. *Science*, 374(6572):1237–1241, 2021.

[51] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE, 1994.

[52] Kaitlin N Smith, Gokul Subramanian Ravi, Jonathan M Baker, and Frederic T Chong. Scaling superconducting quantum computers with chiplet architectures. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1092–1109. IEEE, 2022.

[53] Samuel Stein, Nathan Wiebe, Yufei Ding, James Ang, and Ang Li. Qbeep: Quantum bayesian error mitigation employing poisson modeling over the hamming spectrum. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–13, 2023.

[54] Swamit Tannu, Poulami Das, Ramin Ayanzadeh, and Moinuddin Qureshi. Hammer: boosting fidelity of noisy quantum circuits by exploiting hamming behavior of erroneous outcomes. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS'22)*, pages 529–540, 2022.

[55] Swamit S Tannu and Moinuddin K Qureshi. Mitigating measurement errors in quantum computers by exploiting state-dependent bias. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 279–290, 2019.

[56] Patrick A Truitt, Jared B Hertzberg, CC Huang, Kamil L Ekinci, and Keith C Schwab. Efficient and sensitive capacitive readout of nanomechanical resonator arrays. *Nano letters*, 7(1):120–126, 2007.

[57] Ewout Van Den Berg, Zlatko K Minev, and Kristan Temme. Model-free readout-error mitigation for quantum expectation values. *Physical Review A*, 105(3):032620, 2022.

[58] Vlatko Vedral and Martin B Plenio. Entanglement measures and purification procedures. *Physical Review A*, 57(3):1619, 1998.

[59] Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, Cheng Guo, Chu Guo, Shaojun Guo, Lianchen Han, Linyin Hong, He-Liang Huang, Yong-Heng Huo, Lipeng Li, Na Li, Shaowei Li, Yuan Li, Futian Liang, Chun Lin, Jin Lin, Haoran Qian, Dan Qiao, Hao Rong, Hong Su, Lihua Sun, Liangyuan Wang, Shiyu Wang, Dachao Wu, Yu Xu, Kai Yan, Weifei Yang, Yang Yang, Yangsen Ye, Jianghan Yin, Chong Ying, Jiale Yu, Chen Zha, Cha Zhang, Haibin Zhang, Kaili Zhang, Yiming Zhang, Han Zhao, Youwei Zhao, Liang Zhou, Qingling Zhu, Chao-Yang Lu, Xiaobo Peng, Chen-Zhi adn Zhu, and Jian-wei Pan. Strong quantum computational advantage using a superconducting quantum processor. *Physical review letters*, 127(18):180501, 2021.

[60] Lei Xie, Jidong Zhai, ZhenXing Zhang, Jonathan Allcock, Shengyu Zhang, and Yi-Cong Zheng. Suppressing zz crosstalk of quantum computers through pulse and scheduling co-optimization. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'22)*, pages 499–513, 2022.

[61] Bo Yang, Rudy Raymond, and Shumpei Uno. Efficient quantum readout-error mitigation for sparse measurement outcomes of near-term quantum devices. *Physical Review A*, 106(1):012423, 2022.

[62] Xu Zhang, Wenjie Jiang, Jinfeng Deng, Ke Wang, Jiachen Chen, Pengfei Zhang, Wenhui Ren, Hang Dong, Shibo Xu, Yu Gao, Feitong Jin,

Xuhao Zhu, Qiujiang Guo, Hekang Li, Chao Song, Alexey V. Gorshkov, Thomas Iadecola, Fangli Liu, Zhe-Xuan Gong, Zhen Wang, Dong-Ling Deng, and H Wang. Digital quantum simulation of floquet symmetry-protected topological phases. *Nature*, 607(7919):468–473, 2022.

[63] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun ZHANG, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.