General error mitigation for quantum circuits

Manpreet Singh Jattana[1,2] · Fengping Jin[1] · Hans De Raedt[1,3] ·
Kristel Michielsen[1,2]

Abstract

A general method to mitigate the effect of errors in quantum circuits is outlined. The
method is developed in sight of characteristics that an ideal method should possess
and to ameliorate an existing method which only mitigates state preparation and mea-
surement errors. The method is tested on different IBM Q quantum devices, using
randomly generated circuits with up to four qubits. A large majority of results show
significant error mitigation.

Keywords General error mitigation · Requirements for ideal error mitigation ·
Calibration circuits

1 Introduction

The road of developing and operating devices that would enable quantum computa-
tion has been and continues to be full of obstacles. While some of the development
obstacles had been pointed early on, i.e. implementation of reversibility [1] and loss of
coherence [2], some are found as we progress. Despite these, devices had been devel-
oped and small problems have been implemented [3]. The implementations bring with
them operational obstacles. One operational obstacle is the presence of both known
(e.g. noise, decoherence) and unknown causes that render the computation erroneous.
To tackle this obstacle to some extent, error correction had been proposed [4–6]. Error
correction is difficult to implement on current devices due to its hardware requirements.
Another attempt at removing the erroneous computation obstacle is error mitigation.
In this article, we propose and test a new method for it.

✉ Manpreet Singh Jattana
    m.jattana@fz-juelich.de

1   Institute for Advanced Simulation, Jülich Supercomputing Centre, Forschungszentrum Jülich,
    52425 Jülich, Germany

2   RWTH Aachen University, 52062 Aachen, Germany

3   Zernike Institute for Advanced Materials, University of Groningen, 9747 AG Groningen, The
    Netherlands

1.1 Correction versus mitigation

Error correction has been described as a procedure of protecting quantum computation against noise or errors, by encoding redundant information to the process [7]. This redundancy necessitates additional hardware in order to be implementable. If we wish to avoid using additional hardware, we need a different method altogether.

If we define 'cooperative' as the ability of a quantum device to systematically reproduce its errors, then, *error mitigation* is defined as a method to attenuate errors when applied to a cooperative error-prone device. Since such a process does not require additional hardware, but only additional resources from the available hardware, the main problem that error correction faces is circumvented.

Henceforth, we focus on error mitigation.

1.2 Requirements

We require a method to have the following characteristics to be called an ideal error mitigation method:

1. Result recovery: the method should be able to mitigate errors to a satisfactory accuracy.
2. Depth independence: the method should not depend on circuit depth.
3. Error model: the method should take into account all types of errors a device may be prone to, and not rely on prior information about errors the device is prone to.
4. Practically realisable: the method should make use of resources that are practically similar to the resources used by the circuit which is to be (error) mitigated.
5. No additional hardware: the method should not require additional quantum hardware for being implementable.
6. Gate-set independence: the method should take into account and be applicable to all kinds of quantum gates.
7. No output knowledge: the method should not make use of any specific knowledge about the output of a given circuit.

1.3 Mitigation methods

Various error mitigation methods have been introduced recently. These include methods for error extrapolation and probabilistic error mitigation [8–11], methods that utilise symmetries in circuits and use it for certain errors elimination [12–14], and ideas like quantum subspace expansion [15] or decoherence-free subspaces [16], quantum process tomography [17–20], gate set tomography [21–23], and quasi-probability decomposition [24]. There are methods that focus on read-out error mitigation based on detector tomography [25]. Methods have been tested on trapped-ion devices [26] and superconducting devices [24,27].

We focus on extending the error mitigation method for state preparation and measurement (SPAM) errors mitigation (SPAMEM) available in the IBM's Qiskit library [28]. This method fulfils requirements 2, 5, and 7 and uses a matrix-based approach to mitigate errors. We confine ourselves to comparing the herein proposed method with

Qiskit's standard method. The herein proposed method satisfies the requirements 2, 3, 5, 6, and 7. It satisfies 1 in most of the cases we tested. Requirement 4 is discussed later (see Sect. 4.2.5). This new method builds upon Qiskit's method and is not limited to SPAM but mitigates all kinds of errors.

2 Qiskit error mitigation

Two types of errors produced in any error-prone device can be assumed to be coming from SPAM. If a large part of an erroneous device output is due to SPAM, which is not known a priori, SPAMEM is a useful method. Under the assumption that SPAM errors for a given circuit C_g will also occur for other circuit(s) C_c, we may mitigate them by measuring the effects produced by SPAM errors if the outputs of C_c were known. We call, therefore, C_c a calibration circuit.

Assume that we have an error-prone device which produces some relative frequencies $(v_1, v_2, \ldots, v_{2^N})$, which differ from the ideal (exact) data $(e_1, e_2, \ldots, e_{2^N})$. Later on, when applying error mitigation, we will require a notation for mitigated data $(x_1, x_2, \ldots, x_{2^N})$ and simulator data $(s_1, s_2, \ldots, s_{2^N})$ as well, so let us construct the column vectors

$$V = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{2^N} \end{pmatrix}, \quad E = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_{2^N} \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2^N} \end{pmatrix}, \quad S = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_{2^N} \end{pmatrix}. \tag{1}$$

All these vectors are normalised, i.e. $\sum v_i = \sum e_i = \sum x_i = \sum s_i = 1$, where $i = 1, \ldots, 2^N$. Now, we postulate the existence of a $2^N \times 2^N$ matrix M such that

$$ME = V. \tag{2}$$

Equation (2) serves as a good starting point to understand the basic idea of the method. Note that if the device is not error prone, then M is the identity matrix. For an error-prone device, M has nonzero off-diagonal elements. The purpose of a mitigation method, in the context of this article, is to propose a procedure to be performed on the device, take the outputs to be put in M and use it for error elimination. In the next section, we outline the procedure.

2.1 Calibration and mitigation

To start, we fix the number of qubits and denote it by N. We will require 2^N circuits to be run for the mitigation. The procedure is as follows:

1. Prepare the qubits in all possible 2^N states and measure each state. Each such circuit is a calibration circuit.
2. Enter the data from each calibration circuit into columns of M (and rename it M_Q), where the jth column, starting from left, takes data from the circuit whose state is

given by the binary representation of j, for all $j = 1, \ldots, 2^N$. We refer to M_Q as the *Qiskit calibration matrix*.

Now that we have M_Q, let us use this to mitigate (errors in) the relative frequencies V for a circuit C_g. Instead of Eq. (2), we now have $M_Q X = V$, where X is the mitigated data and may not be always exact (i.e. equal to E). When we proceed to solve for X, we face a problem. Since we did not take into account the fact that we are dealing with relative frequencies which are constrained to be in the interval [0,1], by simply using the inverse of M_Q to solve for X, we may have values in X outside this interval [25,29]. To avoid this problem, we use least squares. Thus, we change our problem to finding the minimum of the function

$$f(x) = \sum_{i=1}^{2^N} (v_i - (M_Q \cdot X)_i)^2, \tag{3}$$

given the constraints $0 \leq x_i \leq 1$ and $\sum x_i = 1$ for all $i = 1, \ldots, 2^N$. For our experiments, we initialize X randomly and use the *minimize* package of *scipy* [30] and the *Sequential Least SQuares Programming* (SLSQP) [31] method.

It is important to note that although Eq. (2) cannot be used in some cases, using Eq. (3) is always possible. This makes the mitigation method a useful heuristic [29].

2.2 Discussion

We now have the error mitigated data X and if we want to test the method, we can compare it to the data S produced by a simulator [see Eq. (1)]. Let us introduce the root mean square errors

$$\Delta X = \sqrt{\sum_{k=1}^{2^N} (x_k - s_k)^2} \text{ and } \Delta V = \sqrt{\sum_{k=1}^{2^N} (v_k - s_k)^2}. \tag{4}$$

If we further define $\Delta_Q = \Delta V - \Delta X$, we can have the following possibilities:

$$\Delta_Q \begin{cases} > 0, & \text{for positive mitigation.} \\ < 0, & \text{for negative mitigation.} \\ = 0, & \text{for no mitigation.} \end{cases} \tag{5}$$

According to requirement 1 from Sect. 1.2, Δ_Q should be positive for all experiments. Furthermore, the level of 'satisfaction' addressed in requirement 1 can be quantified in terms of Δ_Q. For a positive mitigation to be perfect, $\Delta X = 0$ when $\Delta V \neq 0$.

Applying this method on the real device for small depth ($D \approx 2$) circuits works well (data not shown). For larger circuit depths, where significant errors can come from gate operations, Qiskit error mitigation does not improve the results considerably, as

shown later (see Sect. 4.2.1). In the next section, we therefore propose a general error mitigation method.

3 General error mitigation (GEM)

If we add some gates to the preparation and measurement circuit, and the gate operations are error prone, SPAM errors are no longer the only sources of errors. In cases where SPAM errors are not dominant, the Qiskit mitigation approach is unsuitable. In general, we require to mitigate effects of errors that arise not only due to SPAM, but potentially also due to other sources, including but not limited to (for example) erroneous gate operations. If we further postulate that we do not know a priori what sources will contribute to or dominate the errors, as might be the case for any practical quantum computation, we need to include all potential sources of errors. To this end, we need a general method to mitigate errors in a quantum circuit.

For a circuit of any given depth, gate-set, and number of qubits, errors can come from multiple sources. While some sources may be identified and modelled, some others remain unidentified. The identification and the accurate modelling of the error sources present a major challenge. For the general mitigation method, we do not attempt to take this challenge and take a different route, as explained below.

Recollecting our assumption from the beginning of Sect. 2 and applying it to a general case we conclude: under the assumption that all errors in a given circuit C_g will also occur for other circuit(s) C_c, we may mitigate them by measuring the effects these errors produced if the outputs of C_c were known. Using this approach, we avoid modelling individual errors in a circuit altogether and work directly with the device.

For consistency, we borrow Eqs. (1) and (3) and proceed directly by proposing the procedure to be performed on the device to produce M.

3.1 Calibration and mitigation

Assume that we wish to mitigate errors in a given circuit C_g of depth D and with N qubits. The procedure is as follows:

1. Prepare calibration circuits in all possible 2^N states, twice.
2. Consider all the gates of the circuit C_g up to depth $D/2$ (if D is even) or $(D-1)/2$ (if D is odd) and add them to the calibration circuits.
3. Add *inverse gates* of the gates added in step 2, in a reverse order.
4. Measure the calibration circuits and record the data in calibration matrix M_1, similarly as done in Sect. 2.1 step 2.
5. Repeat steps 2, 3, and 4 for the remaining half of the gates on the remaining calibration circuits, and name the new matrix M_2.
6. Calculate the matrix $M_G = (M_1 + M_2)/2$.

Now, we may proceed by using Eq. (3) to find X by simply replacing M_Q by M_G.

Fig. 1 Circuit showing a **H** (Hadamard) and **Y** gate followed by a measurement in the Z basis
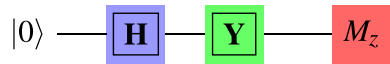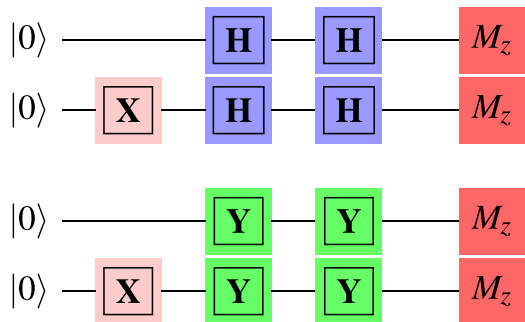

Fig. 2 Four different calibration circuits for the circuit shown in Fig. 1. We prepare the circuits twice in the states $|0\rangle$ and $|1\rangle$ and construct identity circuits by splitting the circuit in half and adding each half together with its inverse (see text)

3.2 Example

To illustrate the GEM procedure, let us consider a simple circuit with one qubit, as shown in Fig. 1. The corresponding calibration circuits are depicted in Fig. 2. The outputs obtained from measuring the calibration circuits are then put into matrices M_1 and M_2.

$$M_1 = \begin{pmatrix} v_1 & v_3 \\ v_2 & v_4 \end{pmatrix}, \quad M_2 = \begin{pmatrix} v_5 & v_7 \\ v_6 & v_8 \end{pmatrix}. \tag{6}$$

The left column of M_1 in Eq. (6) is (the output obtained) from the first circuit in Fig. 2, the right column from the second, the left column of M_2 from the third, and the right column from the last circuit, respectively. The final matrix M_G is then simply the average of M_1 and M_2.

4 Experimental results

4.1 Randomised testing

The GEM method is applied to randomly generated circuits for a large number of cases. Although the method is independent of the gate-set used, for practical reasons we apply only a subset of all possible gates. The used gate-set reads: {**Id**, **U1**, **X**, **Y**, **Z**, **H**, **S**, **S**†, **T**, **T**†, **CNOT**}. These gates are transpiled into device compatible gates as shown in Table 1. Note that on all the different devices tested by us, where all gates are decomposed into some elementary gates called "basis gates", we used all the basis gates. The gate-set we used is a universal gate-set. Therefore, the tested quantum circuits are non-trivial. We created a one-dimensional array of all gates in our gate-set and used *NumPy choice* [32] to randomly choose gates to be added to the circuits. To assess the proposed method, we test it on the IBM Q devices for different numbers of qubits and circuit depths. The following points were kept in mind:

Table 1 The left columns shows
the gates that were added to the
generated circuits, and the right
column shows their
corresponding basis gates
implemented on the quantum
device

Applied gate	Basis gate
ID	**ID**
U1(θ)	**U1**(θ)
X	**U3**($\pi, 0, \pi$)
Y	**U3**($\pi, \pi/2, \pi/2$)
Z	**U1**(π)
H	**U2**($0, \pi$)
S	**U1**($\pi/2$)
S†	**U1**($-\pi/2$)
T	**U1**($\pi/4$)
T†	**U1**($-\pi/4$)
CNOT(c$\rightarrow$t)	**CNOT**(c$\rightarrow$t)

- Create random circuits respecting the physical connectivity of the qubits on the device. Repeat each circuit and its mitigation, to observe any statistical deviations. Repetitions were limited to ten times for all experiments.
- The depths D for each N were chosen after several attempts of finding the suitable depths for error mitigation, based on the actual device performance.
- Although different devices perform differently, no preference was given to any particular combination of N and D to any device. Total shots for all experiments were kept at the maximum supported by the devices, which was 8192 per experiment. The simulator was run with 819,200 shots. Care should be taken in comparing results for different N and D because they may not have been obtained from the same device.
- We set the *circuit optimization level* in Qiskit to zero, and do not include the measurement (gate) in the circuit depth.

4.2 Results and discussion

Figure 3 shows one example of how the output looks like for one experiment on the simulator, on the device, and after error mitigation. For this example, GEM closely recovers the theoretical (exact) results. In this work, in total, we performed six hundred different experiments each repeated ten times. Results are shown in Figs. 4, 5, 6, 7, 8 and 9, where we plot the average ΔV and ΔX [see Eq. (4)] for the ten repetitions of each experiment along with the maximum and minimum datum obtained. By using error mitigation we are hoping to reduce ΔV, where the new value (called the mitigated value) is given by ΔX. Positive values of ΔV (blue squares) represent an erroneous output, and lower values of ΔX (orange circles) represent a more successful mitigation, where $\Delta X = 0$ is a perfect mitigation giving the exact result. For ease of readability, we plot the data in ascending order of the average ΔV.
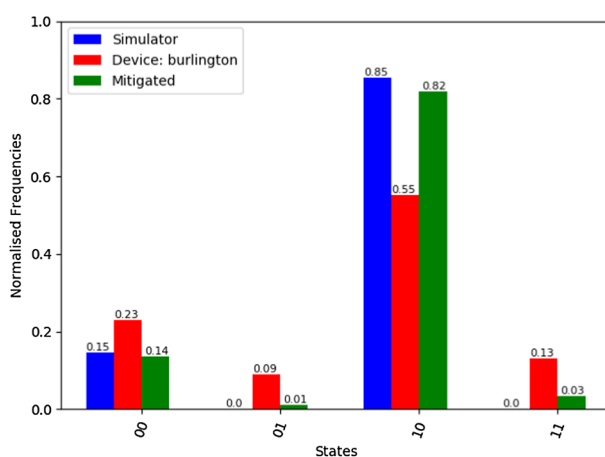
Fig. 3 A sample experiment using the general error mitigation (GEM) method. Shown here are the simulator (left; blue), the device (middle; red), and mitigated (right; green) results for one of the repetitions of experiment number 79 in Fig. 6 (Color figure online)
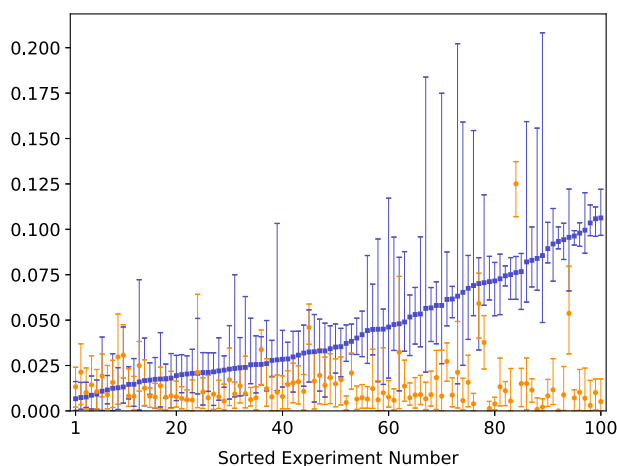

Fig. 4 The average of ΔV (blue square points) and the corresponding average of ΔX (orange circular points) for 100 random circuits, each repeated 10 times, with $N = 1$ (qubit) and depth $D \in [20, 29]$ with mean depth $\bar{D} = 24.73$. The top and bottom of the error bars for ΔV and ΔX represent the maximum and minimum obtained in 10 repetitions, respectively. The device used was IBM Q Armonk [33]. Data are presented in ascending order of average ΔV for easy readability (Color figure online)

In Figs. 4 and 5, we performed GEM on one qubit and different mean circuit depths $\bar{D} \approx 25$ and $\bar{D} \approx 99$, respectively. The effect of increased depth is visible in Fig. 5 where values of ΔV are much larger than those in Fig. 4. This was expected since a larger number of error-prone gates contribute more errors in total. Note that in both cases the error bars are large, suggesting that the output of an experiment repeated ten times fluctuates significantly.
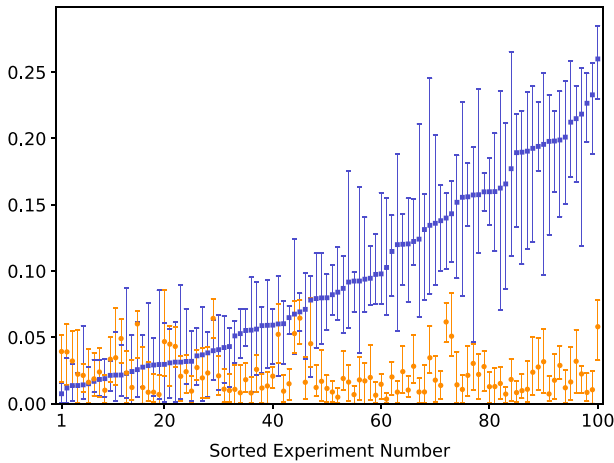
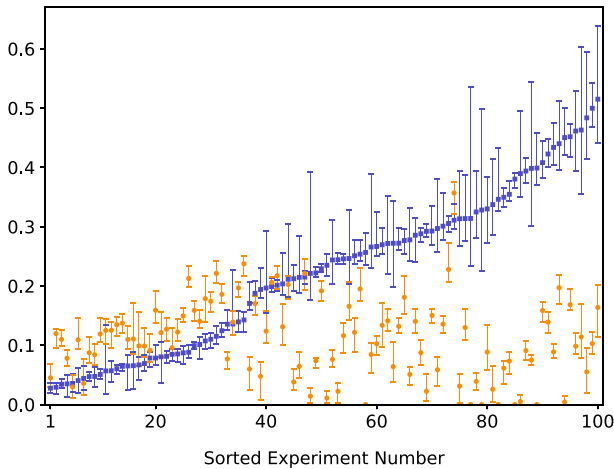**Fig. 5** Same as Fig. 4 except for $D \in [89, 112]$ with $\bar{D} = 99.26$

Fig. 6 Same as Fig. 4 except for $N = 2$ and $D \in [16, 20]$ with $\bar{D} = 19.50$. The device used was IBM Q Burlington [34]

In Figs. 6 and 7, we performed GEM on two qubits and different mean circuit depths $\bar{D} \approx 20$ and $\bar{D} \approx 79$, respectively. In the $\bar{D} \approx 20$ case, negative mitigation occurs significantly more often and this is addressed in the forthcoming subsection (see Sect. 4.2.2). In the $\bar{D} \approx 79$ case, we deliberately remove the **H** gate from the gate set, so that we have outputs that yield only one state, where the effect of errors is most pronounced. We observe positive mitigation for all experiments.

In Fig. 8, we performed GEM on circuits with three qubits and mean circuit depth $\bar{D} \approx 8$. We see that to observe similar ΔV for higher qubit numbers, the depth has to be decreased accordingly and sometimes significantly. This is also visible in Fig. 9, where we performed GEM on circuits with four qubits.
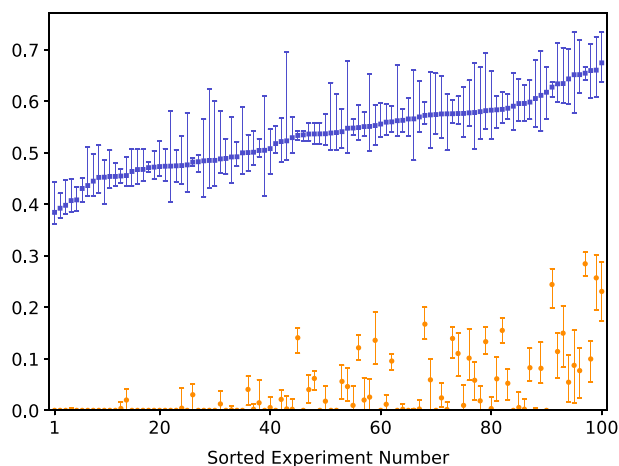
Fig. 7 Same as Fig. 4 except for $N = 2$ and $D \in [74, 80]$ with $\bar{D} = 79.38$. The **H** gate was not used in the gate-set. The device used was IBM Q Burlington [34]
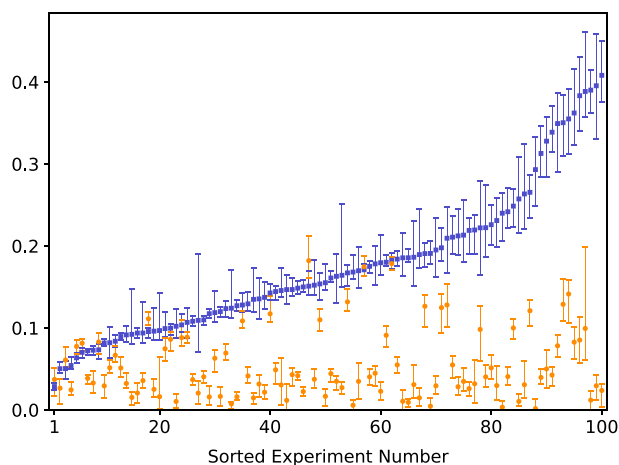

Fig. 8 Same as Fig. 4 except for $N = 3$ and $D \in [6, 10]$ with $\bar{D} = 8.32$. The device used was IBM Q Ourense [35]

We borrow the definitions from Eq. (4) and define $\Delta_G = \Delta V - \Delta X$ for the general case. Also for GEM, Δ_G covers all the possibilities given in Eq. (5). Let us now analyse the data in terms of Δ_G. We choose all $\Delta_G < |0.03 \times \max(\text{average}(\Delta V))|$, or 3% of largest average deviation, to be $\Delta_G \approx 0$. Table 2 shows the number of experiments, corresponding to each N and $\bar{D}$, for which we observed positive, negative, or no mitigation. For most experiments, positive mitigation is seen.
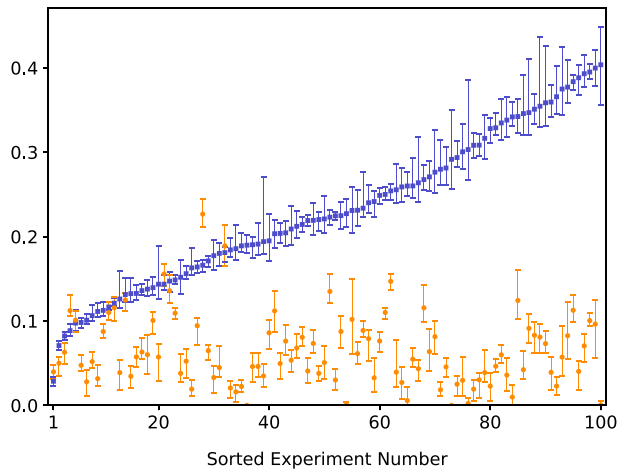
Fig. 9 Same as Fig. 4 except for $N = 4$ and $D \in [6, 10]$ with $\bar{D} = 7.06$. The device used was IBM Q Ourense [35]

Table 2 The distribution of Δ_G observed in the experiments for different qubits and mean depths (see caption of corresponding figure). $\Delta_G > 0$ is positive mitigation, $\Delta_G < 0$ is negative mitigation, and $\Delta_G \approx 0$ is no mitigation

Figures	$\Delta_G > 0$	$\Delta_G < 0$	$\Delta_G \approx 0$
4	85	11	4
5	81	12	7
6	60	32	8
7	100	0	0
8	91	3	6
9	90	2	8

4.2.1 GEM versus QEM

We briefly compare results from general error mitigation (GEM) and Qiskit error mitigation (QEM). We limit our comparison to circuits with two qubits. Results obtained using QEM are shown in Fig. 10. They can be directly compared to those shown in Fig. 7 for which GEM was used. The improvement in error mitigation using GEM, in contrast to QEM, is substantial.

4.2.2 Negative mitigation

Negative mitigation refers to the cases in which the device output is better than the mitigated output. Although Table 2 shows that a majority of experiments indicate positive mitigation, cases of negative mitigation are also present. In Fig. 6, it was observed that experiments with numbers ranging between 1 and 40 correspond to negative mitigation and are comparatively large in number. Owing to this, we restrict our discussion about cases with negative mitigations to the experiments shown in Fig. 6, but it also applies to circuits with other qubit numbers and mean depths.
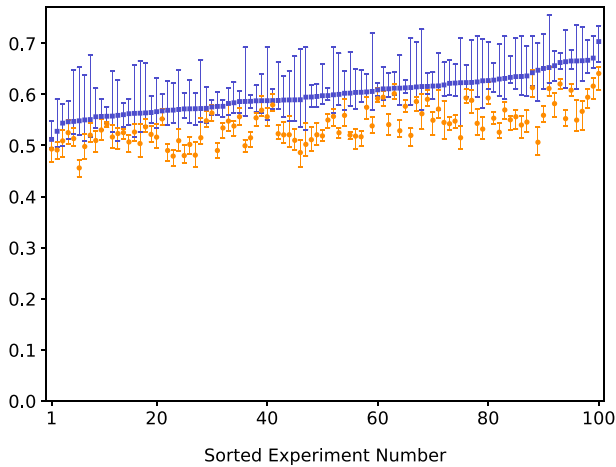
Fig. 10 Same as Fig. 7 except that Qiskit error mitigation was used for $D \in [72, 80]$ with $\bar{D} = 77.50$. The device used was IBM Q Burlington [34]

For a two-qubit circuit, there are only four possible output states. Measurement outputs thereof, range from all normalised frequencies in only one of the states to frequencies equally distributed over all states. As it is reasonable to expect more errors as the depth of a circuit increases, the actual rate depends on various factors, inter alia, the device performance. With an increasing number of errors, the device generates data that deviates more and more from the theoretical results, ultimately producing output states having almost the same frequency (≈ 0.25). Thus, if we intended to have an output equally distributed ($= 0.25$) over all states, errors will not drive the frequencies of the output states too far away from the intended ones, as compared to cases that have only one state as output. When we wish to mitigate errors in the former case, and use C_c that give independent states, the device will be unable to perform well. Then, M_G will not be able to mitigate errors significantly because the errors, although present, did not produce sufficient error effects (relevant for GEM) in the first place. By inspecting the simulation results (not shown), we see that almost all experiments that show negative mitigation have one thing in common, namely, that their circuit produces output states with equal probabilities. Note that the generated circuits were random, and no control was exercised over what the outcomes should be.

From the aforementioned discussion and all the available experimental data, it is discernible that negative mitigation mostly appears in cases where the device is giving outputs V close to the expected ones E (i.e. small ΔV). In such cases, mitigation may not be required at all. Since GEM is a post-processing method, this flexibility offers a user discretion over the need of its use.

Future works need to address the ability of GEM to predict beforehand the cases where mitigation is not required, thereby saving resources. In such undertakings, M_G may be helpful.
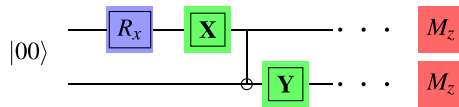
Fig. 11 Circuit (C_g) showing a R_x gate of $\pi/6$ on qubit 1. The gate sequence {**X** gate on qubit 1, **CNOT** gate with control at qubit 1 and target at qubit 2, and **Y** gate on qubit 2} is repeated 30 times followed by a measurement in Z basis

4.2.3 Calibration circuits

Here, we address the question: Is it possible to prepare different calibration circuits? We saw that GEM as a method is able to mitigate errors, given that some assumptions are satisfied. We defined $M_G = (M_1 + M_2)/2$ as the final calibration matrix. Before examining other possibilities, let us briefly discuss the motivations to do so.

If we want to reproduce errors occurring in a given circuit C_g, we should (in the ideal case) use C_g also as the calibration circuit(s), i.e. $C_c \approx C_g$. We use the approximate sign to indicate that the number of gates in both circuits is almost the same and it originates from the fact that we sometimes need an extra gate for the state preparation in C_c, and so the depth may increase. Such an approach, however, is not possible because we do not know a priori the outputs of C_g. A circuit similar to C_g for which the outputs are always known is simply $C_{2c} \approx C_g^{\dagger} C_g$ (identity circuit). This doubles the depth that is why we name it C_{2c}. Thus, by using this approach we are forced to approximate the errors produced in C_g of depth D with errors produced in C_{2c} of depth $2D$. Working with noisy intermediate scale quantum (NISQ) devices like those of IBM, we find that circuits with different depths exhibit different errors. Then, doubling the depth is likely to make the assumption of Sect. 2 inapplicable. For this reason, we cut C_g in half and then take $C_c \approx C_{g/2}^{\dagger} C_{g/2}$ in order to keep the depth(s) of the calibration circuit(s) C_c and the given circuit C_g very close.

Other possible calibration circuit(s) seem to be cases where C_g produces (known) output(s) which are linearly independent. In such cases, we could directly use $C_c \approx C_g$. While GEM can be applied to any circuit, it is still always possible to use any other C_c, the gates in which could be completely different from those in C_g. However, we cannot expect Δ_G for such cases to be positive. We now come back to the question raised above, and answer it in the affirmative, by proposing the following two methods:

(I) If the outputs of C_g are known (or could be known) to be linearly independent, the best strategy may be to use $C_c \approx C_g$.

(II) If a set (or sets) of gates (say $\{g_1, g_2, \ldots, g_p\}$) in C_g are known to produce some errors which can be reproduced by another set of gates (say $\{g_1^c, g_2^c \ldots, g_q^c\}$), such that $q < p$, we may reduce the depth of C_c to be less than depth of C_g by an amount $q - p$.

To show the flexibility of GEM, we present two simple examples where (I) or (II) may help. Consider the two-qubit circuit shown in Fig. 11.

We find that because of the presence of the R_x gate in C_g, calibration circuits do not (theoretically) give linearly independent states in the output. Now, since the number of all other gates is large, the error contribution from R_x may be ignored and we can
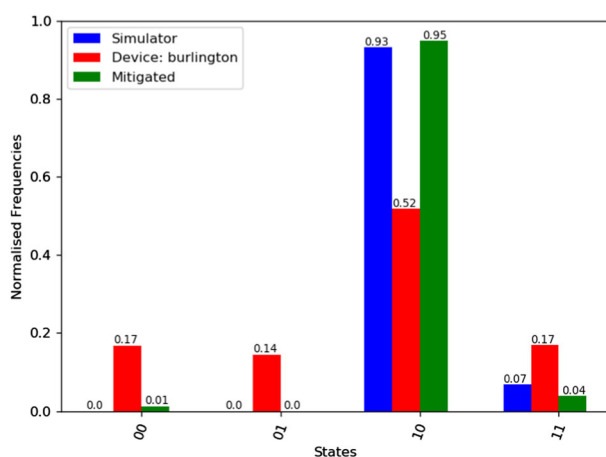
Fig. 12 Simulator (left; blue), device (middle; red), and mitigated (right; green) results for the circuit shown in Fig. 11. The device used was IBM Q Burlington [34] (Color figure online)

remove it altogether from C_c. Doing so will enable us to fill all the columns of M with circuits that produce independent states. Note that by following this new procedure, we abandon requirement 7. The results for this example are given in Fig. 12. We see that the mitigated data are close to the theoretical (simulator) results. The calibration matrix for this experiment:

$$M = \begin{pmatrix} 0.5526123 & 0.1893310 & 0.1623535 & 0.1437988 \\ 0.1372070 & 0.5322266 & 0.1494141 & 0.1748047 \\ 0.1693115 & 0.1330566 & 0.5349121 & 0.1687012 \\ 0.1408691 & 0.1453857 & 0.1533203 & 0.5126953 \end{pmatrix}. \tag{7}$$

In cases where the total number of gates which drive the circuit output in linearly dependent states (like the R_x gate in the previous example) is significant in comparison with the total number of gates, method (II) may help. As an example, we consider the circuit in Fig. 11, but we replace each **Y** gate by an R_y gate. In such a case, if the errors produced by, say, thirty R_y gates, can be satisfactorily reproduced by, say, ten **Z** gates, the thirty R_y gates may be replaced by ten **Z** gates, placed anywhere on qubit 2, under the assumption that an erroneous gate operation produces errors independent of its position in the circuit.

If we restrict ourselves to a device and a limited 'universal' gate-set, and pre-identify gates that closely reproduce the errors produced by multiple other gates, error mitigation may be made quicker. Future work is necessary to address this issue.

4.2.4 Calibration matrix

The calibration matrix plays a central role in both QEM and GEM. Both the methods differ insofar the ways they design the circuits that fill the matrix. This matrix can also offer other insights, as follows. QEM and GEM can work for arbitrary long circuits

if the device produces distinguishable column entries in the calibration matrix. This can be seen by observing the entries within each column of the matrix. If the entries are all nearly equal, then the device was working completely randomly and the output was uniform over all states, and we cannot expect positive mitigation. Alternatively, if the matrix's columns contain distinguishable entries, error mitigation is possible, as seen from the matrix in Eq. (7) and the corresponding positive mitigation in Fig. 12.

4.2.5 Reducing resource consumption

We now look at requirement number 4 as listed in Sect. 1.2 and ask if it is possible to implement GEM using fewer resources. GEM (QEM) requires 2^{N+1} (2^N) calibration circuits to be run when applied to N qubits, in order to fill the matrix M_G (M_Q). As postulated, this M_G contains all information about the errors that a device is prone to. We may choose to approximate the errors to be reproduced in C_c circuits using circuit set $\{C_1, C_2, \ldots, C_p\}$, where $p < 2^{N+1}$ and $c = 2^{N+1}$. This will give us a matrix M_G^*, which will have $c - p$ unfilled columns, to be filled either using the available information from p circuits or with ones on the diagonal and zeros on the off-diagonal elements.

Naturally, such a process may lead to a reduction in cases of significant positive mitigation. We now have fewer empirical entries in M_G^* than in M_G, thereby reducing the information we have about the errors in a given circuit. Thus, there is a trade-off between reducing the resource consumption and increasing the mitigation efficiency. As we use fewer resources, mitigation efficiency decreases, and vice versa.

A possible direction for future work could be to find optimal trade-off criteria when resource consumption is extremely high.

5 Conclusions

We proposed requirements regarding an ideal error mitigation method. Thereafter, we outlined a general error mitigation method applicable to any quantum circuit. This method was tested on different quantum devices, with one to four qubits, using randomly generated circuits. The circuit depths in these tests were limited by the device performance/fidelity. The results showed significant error mitigation. We discussed possibilities to further improve the method.

References

1. Landauer, R., Welland, M.E., Gimzewski, J.K.: Is quantum mechanics useful? Philos. Trans. R. Soc. Lond. Ser. A Phys. Eng. Sci. **353**, 367 (1995)
2. Unruh, W.G.: Maintaining coherence in quantum computers. Phys. Rev. A **51**, 992 (1995)
3. Kandala, A., Mezzacapo, A., Temme, K., Takita, M., Brink, M., Chow, J.M., Gambetta, J.M.: Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature **549**, 242 (2017)
4. Shor, P.W.: Fault-tolerant quantum computation. In: Proceedings of 37th Conference on Foundations of Computer Science, Burlington, VT, USA, pp. 56–65 (1996)
5. DiVincenzo, D.P., Shor, P.W.: Fault-tolerant error correction with efficient quantum codes. Phys. Rev. Lett. **77**, 3260 (1996)
6. Gottesman, D.: Theory of fault-tolerant quantum computation. Phys. Rev. A **57**, 127 (1998)
7. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition, 10th edn. Cambridge University Press, Cambridge (2011)
8. Temme, K., Bravyi, S., Gambetta, J.M.: Error mitigation for short-depth quantum circuits. Phys. Rev. Lett. **119**, 180509 (2017)
9. Li, Y., Benjamin, S.C.: Efficient variational quantum simulator incorporating active error minimization. Phys. Rev. X **7**, 021050 (2017)
10. Endo, S., Benjamin, S.C., Li, Y.: Practical quantum error mitigation for near-future applications. Phys. Rev. X **8**, 031027 (2018)
11. Strikis, A., Qin, D., Chen, Y., Benjamin, S.C., Li, Y.: Learning-based quantum error mitigation. arXiv:2005.07601v1
12. McArdle, S., Yuan, X., Benjamin, S.: Error-mitigated digital quantum simulation. Phys. Rev. Lett. **122**, 180501 (2019)
13. Bonet-Monroig, X., Sagastizabal, R., Singh, M., O'Brien, T.E.: Low-cost error mitigation by symmetry verification. Phys. Rev. A **98**, 062339 (2018)
14. Sagastizabal, R., Bonet-Monroig, X., Singh, M., Rol, M.A., Bultink, C.C., Fu, X., Price, C.H., Ostroukh, V.P., Muthusubramanian, N., Bruno, A., Beekman, M., Haider, N., O'Brien, T.E., DiCarlo, L.: Experimental error mitigation via symmetry verification in a variational quantum eigensolver. Phys. Rev. A **100**, 010302 (2019)
15. McClean, J.R., Kimchi-Schwartz, M.E., Carter, J., de Jong, W.A.: Hybrid quantum-classical hierarchy for mitigation of decoherence and determination of excited states. Phys. Rev. A **95**, 042308 (2017)
16. Premakumar, V.N., Joynt, R.: Error mitigation in quantum computers subject to spatially correlated noise (2018). arXiv:1812.07076
17. Bialczak, R.C., Ansmann, M., Hofheinz, M., Lucero, E., Neeley, M., Oconnell, A.D., Sank, D., Wang, H., Wenner, J., Steffen, M., Cleland, A.N., Martinis, J.M.: Quantum process tomography of a universal entangling gate implemented with Josephson phase qubits. Nat. Phys. **6**, 409 (2010)
18. Howard, M., Twamley, J., Wittmann, C., Gaebe, T., Jelezko, F., Wrachtrup, J.: Quantum process tomography and Linblad estimation of a solid-state qubit. New J. Phys. **8**, 33 (2006)
19. Chow, J.M., Gambetta, J.M., Tornberg, L., Koch, J., Bishop, L.S., Houck, A.A., Johnson, B.R., Frunzio, L., Girvin, S.M., Schoelkopf, R.J.: Randomized benchmarking and process tomography for gate errors in a solid-state qubit. Phys. Rev. Lett. **102**, 090502 (2009). arXiv:0811.4387
20. Neeley, M., Ansmann, M., Bialczak, R.C., Hofheinz, M., Katz, N., Lucero, E., O'Connell, A., Wang, H., Cleland, A.N., Martinis, J.M.: Process tomography of quantum memory in a Josephson-phase qubit coupled to a two-level state. Nat. Phys. **4**, 523 (2008)
21. Merkel, S.T., Gambetta, J.M., Smolin, J.A., Poletto, S., Córcoles, A.D., Johnson, B.R., Ryan, C.A., Steffen, M.: Self-consistent quantum process tomography. Phys. Rev. A At. Mol. Opt. Phys. **87**, 062119 (2013). arXiv:1211.0322
22. Greenbaum, D.: Introduction to quantum gate set tomography (2015). arXiv:1509.02921

23. Blume-Kohout, R., Gamble, J.K., Nielsen, E., Rudinger, K., Mizrahi, J., Fortier, K., Maunz, P.: Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography. Nat. Commun. **8**, 1 (2017). arXiv:1605.07674
24. Song, C., Cui, J., Wang, H., Hao, J., Feng, H., Li, Y.: Quantum computation with universal error mitigation on a superconducting quantum processor. Sci. Adv. **5**, eaaw5686 (2019)
25. Maciejewski, F.B., Zimborás, Z., Oszmaniec, M.: Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography (2019). arXiv:1907.08518
26. Zhang, S., Lu, Y., Zhang, K., Chen, W., Li, Y., Zhang, J.-N., Kim, K.: Error-mitigated quantum gates exceeding physical fidelities in a trapped-ion system. Nat. Commun. **11**, 587 (2020)
27. Kandala, A., Temme, K., Córcoles, A.D., Mezzacapo, A., Chow, J.M., Gambetta, J.M.: Error mitigation extends the computational reach of a noisy quantum processor. Nature **567**, 491 (2019)
28. H.A. et al.: Qiskit: an open-source framework for quantum computing (2019)
29. Geller, M.R.: Rigorous measurement error correction. Quantum Sci. Technol. **5**, 03LT01 (2020). arXiv:2002.01471
30. Jones, E., Oliphant, T., Peterson, P. et al.: SciPy: open source scientific tools for Python (2001)
31. Kraft, D.: A software package for sequential quadratic programming. DLR German Aerospace Center at Institute for Flight Mechanics (1988)
32. Oliphant, T.E.: Guide to NumPy, 2nd edn. CreateSpace Independent Publishing Platform, North Charleston (2015)
33. 1-qubit backend: IBM Q team, "IBM Q 1 Armonk backend specification V1.1.0" (2019)
34. 5-qubit backend: IBM Q team, "IBM Q 5 Burlington backend specification V1.1.4" (2019)
35. 5-qubit backend: IBM Q team, "IBM Q 5 Ourense backend specification V1.0.1" (2019)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.