

Real-time error mitigation for variational optimization on quantum hardware

Matteo Robbiati^{1,2,*}, Alejandro Sopena^{3,4}, Andrea Papaluca^{3,5} and Stefano Carrazza^{1,2,3,6}¹TIF Lab, Dipartimento di Fisica, *Università degli Studi di Milano*, Milan, Italy²Theoretical Physics Department, *CERN*, CH-1211 Geneva 23, Switzerland³Quantum Research Centre, *Technology Innovation Institute*, Abu Dhabi, United Arab Emirates⁴*Instituto de Física Teórica, UAM-CSIC, Universidad Autónoma de Madrid*, Cantoblanco, Madrid, Spain⁵School of Computing, *The Australian National University*, Canberra, ACT, Australia⁶*INFN Sezione di Milano*, Milan, Italy

(Received 13 May 2025; accepted 12 January 2026; published 9 March 2026)

In this work, we put forward the inclusion of error mitigation routines in the process of training variational quantum circuit (VQC) models. In detail, we define a real-time quantum error mitigation (RTQEM) algorithm to assist in fitting functions on quantum chips with VQCs. While state-of-the-art QEM methods cannot address the exponential loss concentration induced by noise in current devices, we demonstrate that our RTQEM routine can enhance VQCs' trainability by reducing the corruption of the loss function. We tested the algorithm by simulating and deploying the fit of a monodimensional u -quark parton distribution function on a superconducting single-qubit device, and we further analyzed the scalability of the proposed technique by simulating a multidimensional fit with up to eight qubits.

DOI: [10.1103/zyb2-zl2d](https://doi.org/10.1103/zyb2-zl2d)

In the era of noisy intermediate-scale quantum (NISQ) [1,2] devices, variational quantum algorithms (VQAs) are the quantum machine learning (QML) models that appear more promising in the near future. They have several concrete applications already validated, such as electronic structure modelization in quantum chemistry [3–6], for instance. Different VQA ansätze have been proposed, such as the Quantum Approximate Optimization Algorithm (QAOA) [7], but they all share as foundation a variational quantum circuit (VQC) consisting of several parametrized gates whose parameters are updated during training.

Hardware errors and large execution times corrupt the landscape in various ways, such as changing the position of the minimum or the optimal value of the loss function, hindering NISQ [1,2] devices' applicability in practice for certain algorithms. Furthermore, VQC models are known to suffer from the presence of noise-induced barren plateaus (NIBPs) [8] in the optimization space, leading to vanishing gradients. NIBPs are fundamentally different from the noise-free barren plateaus discussed in Refs. [9–14]. In fact, approaches designed to tackle noise-free barren plateaus do not seem to effectively address the issues posed by NIBPs [8].

To overcome these limitations, we either have to build fault-tolerant architectures carrying a usable amount of logical qubits or exploit the available NISQ hardware by mitigating

its results from the noise. While the first solution might require significant technical advances, the second one is often achieved with the help of quantum error mitigation (QEM) [15]. Exponential loss concentration cannot be resolved with error mitigation [16], but it is possible to improve trainability by attempting to reduce the loss corruption. Therefore, we define here an algorithm to perform real-time quantum error mitigation (RTQEM) alongside a VQA-based QML training process. A schematic representation of the proposed algorithm is shown in Fig. 1.

In this work, we use the importance Clifford sampling (ISC) method [17], a learning-based quantum error mitigation procedure [18]. The core business of the learning-based QEM techniques is to approximate the noise with a parametric map that, once learned, can be used to clean the noisy results [19–23]. Linear maps have the potential to improve overall trainability by addressing challenges imposed by loss corruptions while not affecting loss concentration itself [16]. The map's parameters are learned during the QML training every time the noise changes above a certain arbitrarily set threshold.

We apply the RTQEM strategy to a series of monodimensional and multidimensional regression problems. First, we train a VQC to tackle a particularly interesting high-energy physics problem: determining the Parton distribution function (PDF) of the u -quark, one of the proton contents. In a second step, we define a multidimensional target to study the impact of the RTQEM procedure when the VQA involves an increasing number of qubits.

Data reuploading [24] is used to encode data into the model, while we implement a hardware-compatible Adam [25] optimizer for the training. We calculate gradients with respect to the variational parameters using the parameter

*Contact author: matteo.robbiati@unimi.it

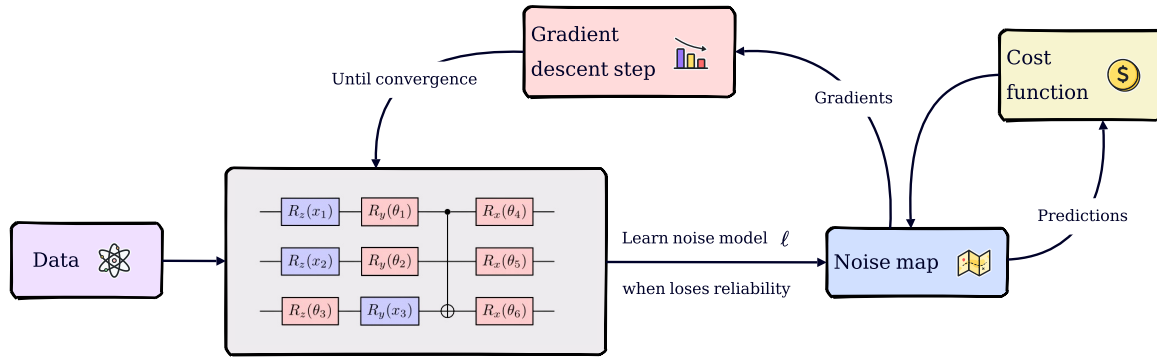


FIG. 1. The RTQEM pipeline involves training a variational quantum circuit on a noisy quantum device using a gradient descent method enhanced with error mitigation. Specifically, the ICS algorithm is used to learn a noise map to mitigate both the gradients and the final predictions. If the noise changes above a certain threshold, the noise map is relearned.

shift rule (PSR) [26,27]. This optimization scheme is ideal for studying the performance of RTQEM, as the PSR formulas require a number of circuits to be executed, which scales linearly with the number of parameters. The greater the number of executions, the better our algorithm must be to enable training of the model.

This setup is then used to perform the full u -quark PDF fit on two different superconducting quantum devices hosted in the Quantum Research Center (QRC) of the Technology Innovation Institute (TII).

The whole work has been realized using the Qibo framework, which offers Qibo [28–31] as high-level language API to write quantum computing algorithms, Qibolab [32–34] as quantum control tool, and Qibocal [35–37] to perform quantum characterization and calibration routines.

The outline is as follows. In Sec. I, we summarize the process of quantum computing with the VQC paradigm, providing also details about the ansatz and the PSR rule we make use of to train the model. In Sec. II, we discuss the impact of noise on the training process. Section III provides an overview of the error mitigation strategy we employed to counteract these effects. Finally, we report the results of our experiments with both noisy simulations and real superconducting qubits' deployment in Sec. IV.

I. METHODOLOGY

A. A snapshot of quantum machine learning

In the following, we are going to consider supervised machine learning problems for simplicity, but what presented here can be easily extended to other machine learning (ML) paradigms in the quantum computation context. Quantum machine learning arises when using quantum computing (QC) tools to tackle ML problems [26,38,39].

In the classical scenario, given an n -dimensional input variable $\mathbf{x}$, a parametric model is requested to estimate a target variable y , which is related to $\mathbf{x} = (x_1, \dots, x_n)$ through some hidden law $y = g(\mathbf{x})$. The model estimations y_{est} are then compared with some measured ground truth data y_{meas} by evaluating a loss function $J(y_{\text{est}}, y_{\text{meas}})$, which quantifies the capability of the model to provide an estimate of the underlying law g . We consider the output variable y as monodimensional for simplicity, but in general it can be mul-

tidimensional. The variational parameters θ of the model are then optimized to minimize (or maximize) the loss function J , leading, in turn, to better predictions y_{est} .

In quantum machine learning, we translate this paradigm to the language of quantum computing. In particular, parametric quantum gates, such as rotations, are used to build VQCs [40], which can be used as parametric models in the machine learning process. Once a parametric circuit $\mathcal{U}(\theta)$ is defined, it can be applied to a prepared initial state $|\psi_0\rangle$ of a quantum system to obtain the final state $|\psi_f\rangle$, which is used to evaluate the expected value of an arbitrary chosen observable $\mathcal{O}$,

$$f(\theta)_{\mathcal{O}} = \langle \psi_0 | \mathcal{U}^\dagger(\theta) \mathcal{O} \mathcal{U}(\theta) | \psi_0 \rangle. \quad (1)$$

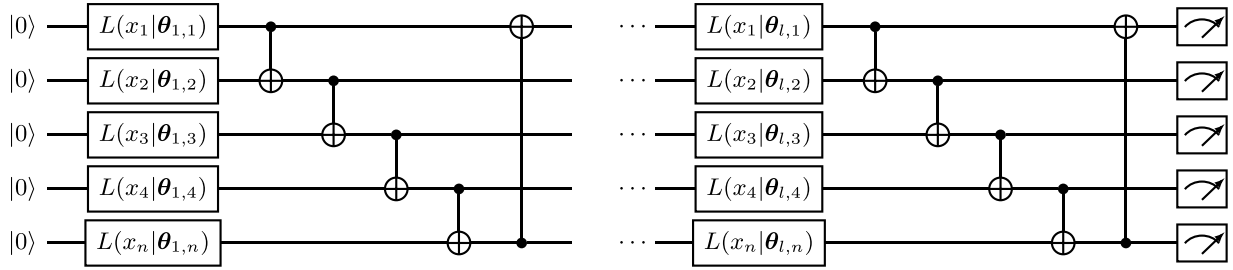
Various methods exist to embed input data into a QML process [41–43]; in this work, we employ the reuploading strategy [24]. The estimates of y can be obtained by calculating expected values of the form (1). Finally, the circuit's parameters are optimized to minimize (or maximize) a loss function J , pushing f as close as possible to the unknown law g .

B. A variational circuit with data reuploading

The data reuploading [24] method is built by defining a parameterized layer made of fundamental uploading gates that accepts the input data $\mathbf{x}$ to be uploaded. Then, the reuploading of the variable is achieved by building a circuit composed of a sequence of uploading layers. Inspired by Ref. [44], we build our ansatz by defining the following fundamental uploading gate,

$$L(x_j | \theta_{l,j}) = R_z(\theta_3 x_j + \theta_4) R_y(\theta_1 \kappa(x_j) + \theta_2), \quad (2)$$

where x_j is the j th component of the input data and with $\theta_{l,j}$ we denote the four-parameter vector composing the gate that uploads x_j at the ansatz layer l . The information x_j is uploaded twice in each L , first in R_y and second in R_z through an activation function $\kappa(x_j)$. To embed the n components of $\mathbf{x}$ into the ansatz, we build a n -qubit circuit $\mathcal{U}$, where the single-qubit gates are the fundamental uploading gates, and entanglement is generated with cnot gates, as shown in Fig. 2. We calculate $f(1)$ as the expected value of the Pauli observable $\sigma_z^{\otimes n}$ on the final state $\mathcal{U}(|0\rangle^{\otimes n})$.


 FIG. 2. Circuit ansatz to reupload the data $\mathbf{x}$ with n qubits and l layers.

C. Gradient descent on hardware

Gradient-based optimizers [25,45–47] are commonly employed in machine learning problems, particularly when using neural networks [48] (NNs) as models. In the QML context, VQCs are utilized to construct quantum neural networks [49], which serve as quantum analogs of classical NNs. Consequently, we are naturally led to believe that methods based on gradient calculation could be effective.

1. The parameter shift rule

In order to perform a gradient descent on an NISQ device, we need a method that is robust to noise and executable on hardware. This cannot be done as in the classical case following a back-propagation [45] of the information through the network. We would need to know the f values during the propagation, but accessing this information would collapse the quantum state. Moreover, standard finite-differences methods are not applicable due to the shot noise when executing the circuit a finite number of times. An alternative method is the so-called parameter shift rule [27,50–53], which enables the evaluation of quantum gradients directly on the hardware [27]. Given f as introduced in Eq. (1) and considering a single parameter $\mu \in \theta$ affecting a single gate whose Hermitian generator has at most two eigenvalues, the PSR allows for the calculation

$$\partial_\mu f = r(f(\mu^+) - f(\mu^-)), \quad (3)$$

where $\pm r$ are the generator eigenvalues, $\mu^\pm = \mu \pm s$ and $s = -\pi/4r$. In other words, the derivative is calculated by executing twice the same circuit $\mathcal{U}(\theta)$ in which the parameter μ is shifted backward and forward of s . A remarkable case of the PSR involves rotation gates, for which we have $r = 1/2$ and $s = \pi/2$ [26].

2. Evaluating gradients of a reuploading model

In order to perform a gradient-based optimization, we first need to calculate the gradient of a loss function J with respect to the variational parameters of the model. Then, the derivatives are used to perform an optimization step in the parameters' space by following the steepest direction of the gradient,

$$\theta_{l+1} = \theta_l - \eta \nabla J(\theta_l), \quad (4)$$

where η is the learning rate of the gradient descent algorithm. Since our QML model is a circuit in which the variational parameters are rotation angles, such derivatives can be estimated by the PSR (3). However, even in the simplest case, this kind

of procedure can be computationally expensive, since for each parameter we need two evaluations of f , as illustrated in Eq. (3). Given a VQC with p variational parameters, a training set size of N_{data} , and a budget of N_{shots} for each function evaluation, the total computational cost amounts to $2pN_{\text{shots}}N_{\text{data}}$ circuit executions. This high number of evaluation is useful for testing the effectiveness of error mitigation routines, which can be applied to every function evaluation of the algorithm. We followed the same optimization strategy described in Refs. [54,55], defining a mean-squared error loss function,

$$J_{\text{mse}}(\mathbf{x}^i | \theta) = \frac{1}{N_{\text{data}}} \sum_i^{N_{\text{data}}} [f(\mathbf{x}^i, \theta) - g(\mathbf{x}^i)]^2, \quad (5)$$

where the superscript denotes the i th variable $\mathbf{x}$ of the dataset. Note that this differs from the subscripts used so far to denote the components of the variable $\mathbf{x}$.

Our total execution time is dominated by the effect of circuit switching and network latency costs rather than shot cost. Therefore, we prefer to reduce the number of iterations at the expense of increasing the number of shots per iteration. In this context, the Adam [25] optimizer stands out due to its robustness when dealing with complex parameter landscapes.

II. NOISE ON QUANTUM HARDWARE

Recognizing the impact of noise on the optimization landscape is crucial in practical quantum computing implementations. In the presence of a general class of local noise models, for many important ansätze such as hardware efficient ansatz, the gradient decreases exponentially with the depth of the circuit d . Typically, d scales polynomially with the number of qubits n , causing the gradient to decrease exponentially in n . This phenomenon is referred to as a noise-induced barren plateau [8]. NIBPs can be seen as a consequence of the loss function converging around the value associated with the maximally mixed state. Furthermore, noise can corrupt the loss landscape in various ways such as changing the position of the minimum.

In order to quantify these effects, we consider a noise model composed of local Pauli channels acting on qubit j before and after each layer of our ansatz,

$$\mathcal{P}_j(\sigma) = q_j \sigma, \quad (6)$$

where $-1 < q_x, q_y, q_z < 1$ and σ denotes the local Pauli operators $\{\sigma_x, \sigma_y, \sigma_z\}$. We represent the channel in the Pauli-Liouville form, i.e., by its action on the Pauli operators rather than in Kraus form. The overall channel is $\mathcal{P} = \bigotimes_j^N \mathcal{P}_j$. We

also include symmetric readout noise $\mathcal{M}$ made of single-qubit bit-flip channels with bit-flip probability $(1 - q_M)/2$. This results in the noisy expectation value,

$$f_{\text{noisy}} = \text{Tr}[\sigma_z^{\otimes n}(\mathcal{M} \circ \mathcal{P} \circ L_l \circ \dots \circ L_1 \circ \mathcal{P})(|\psi_0\rangle\langle\psi_0|)]. \quad (7)$$

The NIBP translates into a concentration of the expectation value around 0 [8],

$$|f_{\text{noisy}}| < 2q_M^n q^{2l+2} \left(1 - \frac{1}{2^n}\right). \quad (8)$$

Certain loss functions exhibit noise resilience; i.e., their minimum remains in the same position under the influence of certain noise models, even though its value may change. Contrarily, our loss function (5) is not noise resistant. We aim to explore the extent to which it is possible to mitigate the noise and enhance the training process of VQCs with nonresistant loss functions.

A. Error mitigation

Recent research [56–61] has focused on developing methods to define unbiased estimators of the ideal expected values leveraging the knowledge about the noise that we can extract from the hardware. However, these estimators are also affected by exponential loss concentration, implying that NIBPs cannot be resolved without requiring exponential resources through error mitigation [16].

In the regime where loss concentration is not severe, it is also not straightforward for error mitigation to improve the resolvability of the noisy loss landscape, thus alleviating exponential concentration.

The variance of the error-mitigated estimators is typically higher than that of the mean estimator [62], setting up a trade-off between the improvement due to bias reduction and the worsening caused by increased variance. Relative resolvability quantifies this trade-off by giving the factor by which the required shot count changes when applying mitigation. For two parameter points θ_1, θ_2 , it can be written as

$$r = \frac{N_{\text{noisy}}}{N_{\text{mit}}} = \frac{\text{Var}(\hat{f}_{\text{noisy}})}{\text{Var}(\hat{f}_{\text{mit}})} \times \frac{[f_{\text{mit}}(\theta_1) - f_{\text{mit}}(\theta_2)]^2}{[f_{\text{noisy}}(\theta_1) - f_{\text{noisy}}(\theta_2)]^2}, \quad (9)$$

where $\hat{f}_{\text{noisy}}$ and $\hat{f}_{\text{mit}}$ denote the noisy and mitigated estimators.

While mitigation techniques such as zero noise extrapolation, probabilistic error cancellation, and virtual distillation are often effective at reducing bias in expectation value estimates, they typically degrade the resolvability of the VQA cost landscape. They either compress the separation between nearby cost values or increase the estimator variance, which reduces resolvability and requires many more shots for reliable optimization [17, 19–23]. In contrast, linear model-based approaches such as Clifford data regression maintain the ratio between cost differences and variance under global depolarizing noise, thereby preserving resolvability even in noisy conditions [16]. Among these methods, we adopt the importance Clifford sampling [17] scheme due to its ability to handle single-qubit dependent noise, its scalability, and its modest resource requirements, relying only on additional Clifford circuits rather than near-Clifford constructions.

1. Importance Clifford sampling

Suppose we want to estimate the expected value of an observable $\mathcal{O}$ for the state ρ prepared by a quantum circuit $\mathcal{C}^0$. In a realistic situation we are going to obtain a noisy expected value $\langle\mathcal{O}\rangle_{\text{noisy}}^0$ different from the true one $\langle\mathcal{O}\rangle^0$. The idea behind importance Clifford sampling (ICS) is to generate a set of m training Clifford circuits $\mathcal{S} = \{\mathcal{C}^i\}_{i=1}^m$ with the same circuit frame as the original one $\mathcal{C}^0$. The classical computation of noiseless expected values of Clifford circuits is efficient [63, 64]. This enables us to compute the ideal expected values $\{\langle\mathcal{O}\rangle^i\}_{i=1}^m$ through simulation, as well as the noisy expected values $\{\langle\mathcal{O}\rangle_{\text{noisy}}^i\}_{i=1}^m$ when evaluating them on hardware.

When $\mathcal{O}$ is a Pauli string, the noise-free expected values will concentrate on $-1, 0$, and 1 [63]. Furthermore, as discussed in Ref. [17], not all the Clifford circuits are error sensitive. In particular, we only need circuits whose expected values on Pauli's are ± 1 . We refer to these circuits as *nonzero* circuits for simplicity. Unfortunately, sampling *nonzero* circuits is exponentially rare when the number of qubits increases; thus, a strategy has to be defined to efficiently build a suitable training set. We follow the ICS algorithm [17], in which *nonzero* circuits are built by adding a layer of Pauli gates to *zero* circuits. These gates can be merged with the ones following so that the depth does not increase.

The generated set is then used to train a model to learn a mapping between $\langle\mathcal{O}\rangle_{\text{noisy}}$ and $\langle\mathcal{O}\rangle$. The structure of the model ℓ can be inspired by considering the action of a global depolarizing channel with depolarizing parameter λ ,

$$\langle\mathcal{O}\rangle_{\text{noisy}} = (1 - \lambda)\langle\mathcal{O}\rangle + \frac{\lambda}{d} \text{Tr}(\mathcal{O}), \quad (10)$$

where $d = 2^n$ denotes the dimension of the Hilbert space and $0 < \lambda < 4^n/(4^n - 1)$. Focusing on Pauli strings and allowing λ to take any value, we arrive at the phenomenological error model,

$$\langle\mathcal{O}\rangle_{\text{noisy}}^i = (1 - \lambda_c^i)\langle\mathcal{O}\rangle^i, \quad (11)$$

from which we can calculate λ_c^i for each circuit in the training set. This set of depolarizing parameters, characterized by the mean value $\lambda_0 = \langle\lambda_c\rangle_{\mathcal{S}}$ and standard deviation σ , allows to define an effective depolarizing parameter for mitigating the initial circuit,

$$\lambda_{\text{eff}} = \lambda_0 - \frac{\sigma^2}{1 - \lambda_0}. \quad (12)$$

This translates into the noise map,

$$\ell(\langle\mathcal{O}\rangle|\lambda_{\text{eff}}) = \frac{(1 - \lambda_0)}{(1 - \lambda_0)^2 + \sigma^2} \langle\mathcal{O}\rangle_{\text{noisy}}. \quad (13)$$

The average depolarizing rate λ_0 scales proportionally with the number of gates, while the standard deviation σ is proportional to its square root [17]. This implies that the model performs better as the circuit depth increases.

The noise map (13) effectively handles symmetric readout noise, but fails with asymmetric noise. For these situations, we employ Bayesian iterative unfolding [65] to mitigate measurement errors in advance.

A schematic representation of the described algorithm is reported in Fig. 3.

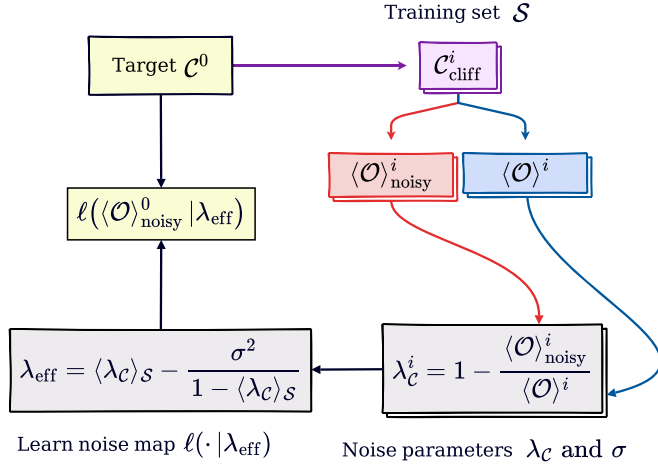


FIG. 3. Importance Clifford sampling is a learning-based error mitigation algorithm that uses a set of Clifford circuits to learn a noise map to mitigate the expected value of a given observable.

III. THE RTQEM ALGORITHM

We implement an Adam optimization mitigating both gradients and predictions following the procedure presented in Sec. II A 1.

In a real quantum computer, small fluctuations of the conditions over time, such as temperature, may result in a change of the shape of the noise sufficient to deteriorate results. Therefore, we compute a metric

$$D(z, \ell(z)) = |z - \ell(z)| \quad (14)$$

at each optimization iteration, which quantifies the distance between a target noiseless expected value z and the mitigated estimation $\ell(z)$. These expected values are calculated over a single *nonzero* test circuit to maximize the bias. If an arbitrary set threshold value ε_ℓ is exceeded, the noise map is relearned from scratch. Since the reliability metric $D(z, \ell(z))$ naturally fluctuates due to shot noise and small variations of the device conditions, the threshold ε_ℓ should be chosen relative to these intrinsic fluctuations. A simple rule of thumb is to set ε_ℓ to be on the order of the standard deviation of the effective depolarizing parameter, i.e.,

$$\varepsilon_\ell \gtrsim \text{std}(\lambda_{\text{eff}}), \quad (15)$$

which avoids triggering unnecessary relearning events.

A schematic representation of the algorithm is shown in Algorithm 1.

IV. VALIDATION

We propose two different experiments to test the RTQEM algorithm introduced above. First, in Sec. IV A, we simulate the training of a VQC on both a single and a multi-qubit noisy device, whereas in Sec. IV B the same procedure is deployed on a superconducting single-qubit chip. The programs to reproduce such simulations can be found in Ref. [66].

ALGORITHM 1 RTQEM.

```

Set  $N_{\text{update}}, N_{\text{epoch}}, k = 0$ ;
Initialize VQC parameters  $\theta_k$ , noise map  $\ell$ ;
Define target noiseless expectation value  $z$ ;
Define metric  $D(z, \ell(z))$  to check  $\ell$  reliability;

for  $k < N_{\text{epochs}}$  do
    if  $D(z, \ell(z)) > \varepsilon_\ell$  then
        learn  $\ell_k$ ;
         $\ell \leftarrow \ell_k$ ;
    end
    compute  $\ell(\mathbf{y}_{\text{est}})$ ;
    calculate  $J[\ell(\mathbf{y}_{\text{est}}), \mathbf{y}_{\text{meas}}]$ ;
    for  $p \in \theta_k$  do
        compute  $\ell(\partial_p J)$  via PSR;
    end
     $\theta_{k+1} \leftarrow \text{Adam}(\theta_k)$ ;
end
    
```

A. Simulation

In this section, we benchmark different levels of error mitigation by conducting both noisy and noiseless classical simulations with $N_{\text{shots}} = 10\,000$ shots as outlined in Table I. The VQC shown in Fig. 2 is used as ansatz and the noise is described by the noise model presented in Sec. II. We first consider a static-noise scenario in Sec. IV A 1, while in Sec. IV A 2 we let the noise vary over time.

1. Static-noise scenario

The following simulations are performed using a static local Pauli noise model where we set the following noise parameters $q_x = 0.007$, $q_y = 0.003$, $q_z = 0.002$, and $q_M = 0.005$.

We first consider a one-dimensional target, namely, the u -quark PDF for a fixed energy scale Q_0 with varying momentum fraction x sampled from the interval $[0, 1]$. A logarithmic sampling is used to improve the resolution of the $x \sim (0, 0.1)$ range where the shape of the function is more rugged. The corresponding PDF values are provided by the NNPDF4.0 grid [67]. We address this first target by constructing a four-layer single-qubit circuit, following the ansatz depicted in Fig. 2. The results, shown in Fig. 4, illustrate that the RTQEM approach enables the training to converge to the correct solution.

To avoid the u -quark PDF comfortably resting below the bound (8) and thereby disguising the effect of the NIBP, we opted to expand it to cover the range $[0, 1]$. One might wonder whether a similar, but opposite, trick could be employed in case that the bound intercepts the target function, therefore compressing the function to make it lie below the bound and

TABLE I. Summary of the tested simulation configurations.

Training	Noise	RTQEM	QEM at the end
Noiseless	×	×	×
Noisy	✓	×	×
fQEM	✓	×	✓
RTQEM	✓	✓	✓

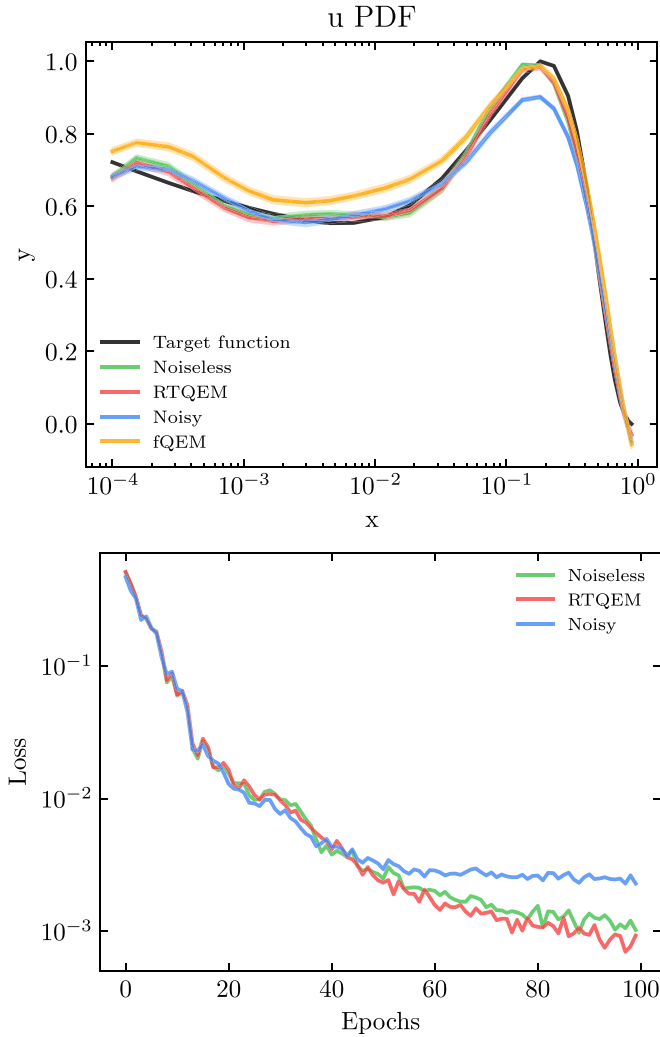


FIG. 4. Estimates of u -quark PDF associated with $N_{\text{data}} = 50$ momentum fraction values sampled logarithmically in $[0, 1]$. The NNPDF4.0 measures (black line) are compared with results obtained through noiseless simulation (green line), noisy simulation (blue line), noisy simulation with mitigation applied to the final predictions (yellow line), and real-time mitigated noisy simulation (red line). The effective depolarizing parameter λ_{eff} is 0.09 ± 0.01 . The final predictions are computed averaging on $N_{\text{runs}} = 100$ predictions calculated for each of the N_{data} points. The confidence intervals are obtained using one standard deviation from the mean. The bottom plot shows the loss function history for each training scenario.

avoid any sort of limitations on the predictions. While this is a perfectly viable method in theory, it is essentially pointless in practice. The compression of the function, indeed, will also increase the precision needed to resolve it, which translates in a larger number of shots required by each prediction [16].

The noisy simulation is clearly limited by loss concentration (8), which caps the predictions at around $y \simeq 0.85$. This limit is also noticeable in Fig. 4, where the loss value cannot decrease below the threshold with unmitigated training. Attempting to correct the predictions posttraining (fQEM) allows access to the region above the bound but does not improve the fit. This is expected, since noise shifts the position of the minima of the loss function, making it impossible to

TABLE II. Mean-squared error distances between the target functions and the VQC fitting model trained under the different conditions of Table I.

Target	MSE _{noiseless}	MSE _{noisy}	MSE _{fqem}	MSE _{rtqem}
u PDF	0.008	0.018	0.023	0.008
$\cos 4d$	0.003	0.043	0.140	0.003
$\cos 6d$	0.002	0.083	0.214	0.002
$\cos 8d$	0.001	0.118	0.360	0.004

recover the true minimum with a final rescaling pass alone. We therefore include fQEM only as a simple baseline to demonstrate that our cost function is not noise resistant; i.e., noise not only alters the value of the cost function at its minimum but can also change its position in parameter space. In this sense, fQEM serves to illustrate this effect rather than to provide a state-of-the-art mitigation method. However, by gradually cleaning up the loss function landscape during the training, the correct minimum is recovered, and the fit converges to the target function. To better understand how the algorithm scales with the number of qubits, we study the problem of fitting a multidimensional function. In particular, we consider

$$f_{\text{ndim}}(\mathbf{x}; \boldsymbol{\beta}) = \sum_{i=1}^{N_{\text{dim}}} [\cos(\beta_i x_i)^i + (-1)^{i-1} \beta_i x_i], \quad (16)$$

where both data $\mathbf{x}$ and parameters $\boldsymbol{\beta}$ have dimension N_{dim} and the index i runs over the problem dimensions. In particular, the model parameters $\boldsymbol{\beta}$ are defined as equidistant point in the range $[0.5, 2.5]$, and they are kept fixed during the optimization. The target f_{ndim} is rescaled in order to occupy the range $[0, 1]$. We consider N_{data} values for each $x_i \in \mathbf{x}$ homogeneously distributed in the range $[0, 1]$. The ansatz is the N_{dim} -qubit circuit of Fig. 2 with three layers.

As the dimensionality of the problem increases, and consequently, the number of qubits, the noise-induced bound is lower, hindering the description of the function in a region of its domain. By applying the RTQEM algorithm, we manage to achieve lower values of the loss function, thereby improving the quality of the fit (see Fig. 5). These results are confirmed by computing the mean-squared error (MSE) metric,

$$\text{MSE} = \frac{1}{N_{\text{data}}} \sum_{j=1}^{N_{\text{data}}} (\bar{y}_{\text{est}}^j - y_{\text{meas}}^j)^2, \quad (17)$$

where $\bar{y}_{\text{est}}^j$ is the average estimate of $f(\mathbf{x}^j)$ over N_{runs} . The MSE associated with each fit is shown in Table II.

Regarding the gradients, it is important to note that there are no significant differences between the raw gradients and the exact gradients (see Appendix A). This means that we are in a regime where the loss concentration is not severe, and there is still room for error mitigation to improve trainability by mitigating other unwanted effects in the landscape due to the noise.

2. Evolving-noise scenario

To study the performance of the method with noise evolution, we consider a random change in the Pauli parameters of

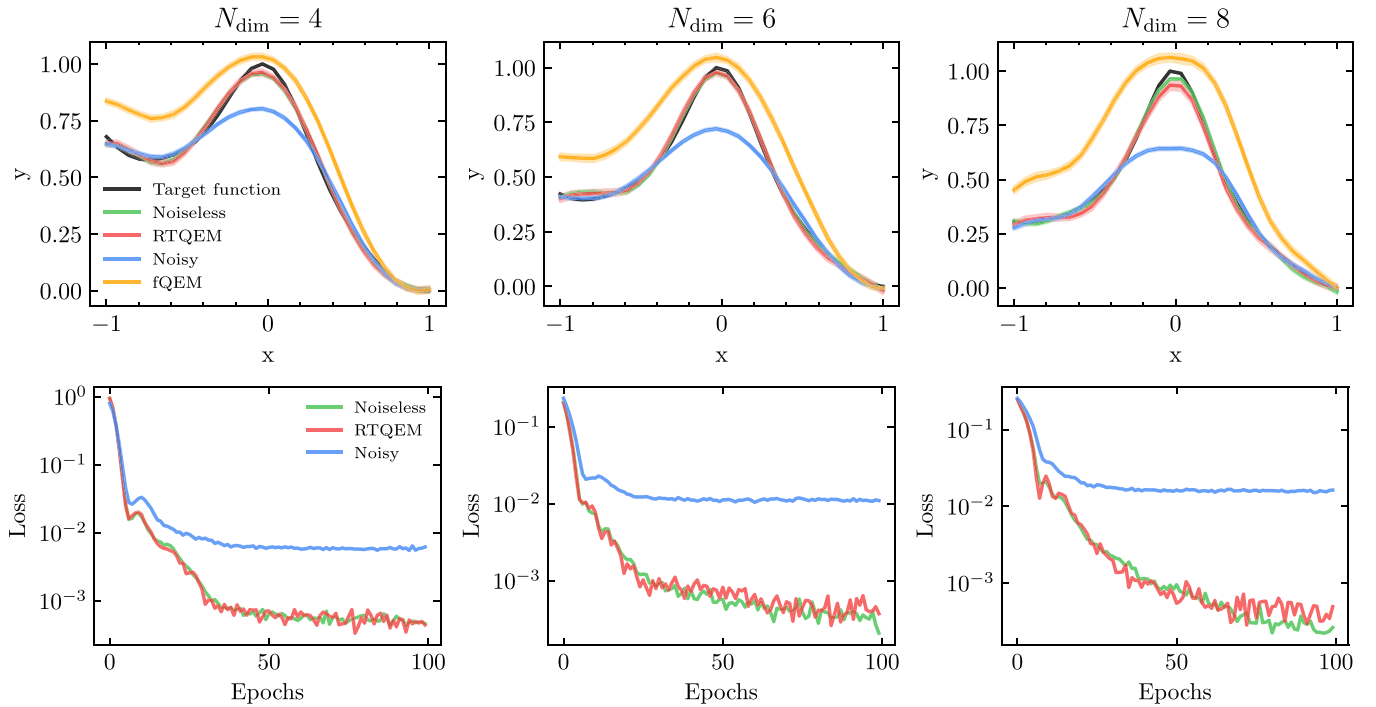


FIG. 5. Predictions for the multidimensional function f_{ndim} with $N_{dim} = 4, 6, 8$ from left to right. The exact predictions (black line) are compared with results obtained through noiseless simulation (green line), noisy simulation (blue line), noisy simulation with mitigation applied to the final predictions (yellow line), and real-time mitigated noisy simulation (red line). The effective depolarizing parameters λ_{eff} are 0.22 ± 0.02 ($N_{dim} = 4$), 0.31 ± 0.03 ($N_{dim} = 6$), and 0.41 ± 0.02 ($N_{dim} = 8$). The final predictions are computed averaging on $N_{runs} = 20$ predictions calculated for each of the $N_{data} = 30$ points. The confidence intervals are obtained using one standard deviation from the mean. The bottom plot shows the loss function history for each training scenario.

the noise model in each epoch. In particular, the initial parameter vector $\mathbf{q}^0 = (q_x^0, q_y^0, q_z^0)$ is moved in its three-dimensional space following a procedure similar to a random walk (RW) on a lattice. Namely, each component q_j is evolved from epoch k to epoch $k + 1$ as

$$q_j^{(k+1)} = q_j^k + r\delta, \quad (18)$$

where $r \sim \{-1, +1\}$ and the step length is sampled from a normal distribution $\delta \sim \mathcal{N}(0, \sigma_\delta)$. We refer to an evolution performing N steps governed by σ_δ as $RW_{\sigma_\delta}^N$. The readout noise parameter is kept fixed at the value of $q_M = 0.005$. In this evolving scenario, when the metric (14) exceeds a certain threshold ε_ℓ , the mitigation parameter λ_{eff} (12) is updated.

To evaluate the effect of relearning the noise on the training process, we track the evolution of the loss function of the u -quark PDF for various values of ε_ℓ , as shown in Fig. 6. We aim for a reduction in the loss function to correspond to a closer approximation to the noise-free parameters. Therefore, we recalculate the loss function values at each iteration using the noisy training parameters, but in a noiseless simulation. As the threshold decreases, the noise map is updated more frequently. It is expected that a lower threshold will enhance the training until it reaches a certain minimum value, characterized by the standard deviation of λ_{eff} . Interestingly, even a few updates to the noise map can lead to significantly lower values for the loss function. For instance, the difference between the minimum values of the loss function when updating the noise map 8 times as opposed to 93 times during a training of 100 epochs

is $\mathcal{O}(10^{-3})$. This suggests that a minor additional classical computational cost can significantly improve the training.

B. Training on hardware

We set up our full-stack gradient descent training on a superconducting device hosted by the QRC in the TII. The high-level algorithm is implemented with Qibo [28–31] and then translated into pulses and executed on the hardware through the Qibolab [32,33] framework (see Appendix B). The qubit calibration and characterization have been performed using Qibocal [35,36]. In particular, we use one qubit of Soprano, a five-qubit chip constructed by QuantWare [68] and controlled using Qblox [69] instruments (see Appendix C). We refer to this device as qw5q.

The u -quark PDF for a fixed energy scale Q_0 is targeted using a four-layer single-qubit circuit built following the ansatz presented above. We take $N_{data} = 15$ values of the momentum fraction x sampled from the interval $[0, 1]$.

An estimate to the bound imposed by the noise is provided by the assignment fidelity of the used qubits, which are collected in dedicated runcards describing the current status of the QRC devices [70]. As with the simulation, we adjust the function to span the range $[0, 1]$. We perform a gradient descent on the better calibrated qubit of qw5q using the parameters collected in Table III. The training has been performed for both the unmitigated and the RTQEM approaches. After training, we repeat $N_{runs} = 10$ times the predictions for each one of $N_{target} = 30$ target values of x sampled logarithmically

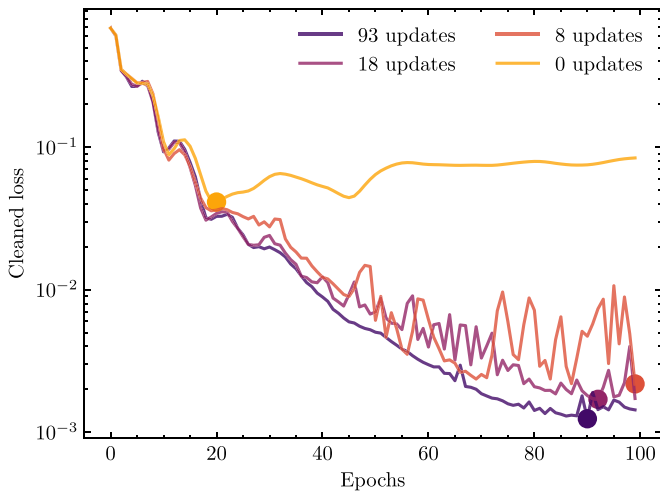


FIG. 6. Four RTQEM training simulations sharing the same initial conditions. The initial local Pauli parameters $\mathbf{q}^0 = (0.005, 0.005, 0.005)$ are evolved under an $\text{RW}_{0.002}^{100}$. The readout noise parameter has been kept fixed to $q_M = 0.005$, four layers are used with $N_{\text{data}} = 30$, and $\eta = 0.05$. For each training simulation, a different noise threshold value was used: $\varepsilon_\ell = \{0, 0.05, 0.1, 0.2\}$. As a result, λ_{eff} is relearned $u = \{93, 18, 8, 0\}$ times, respectively. We show the loss function values computed using the training parameters at each iteration but deployed in a noiseless scenario.

from $[0, 1]$. The final estimate to the average prediction and its corresponding standard deviation are computed out of the N_{runs} repetitions.

The RTQEM process leads to better compatibility overall and, in particular, is able to overcome the bound set by the noise represented as a black horizontal line, as shown in Fig. 7. Indeed, the mitigated fit leads to a smaller MSE compared to the unmitigated one, as reported in Table IV. This proves that the RTQEM procedure gives access to regions that are naturally forbidden to the raw training.

As a second test, we push forward the RTQEM training by performing a longer optimization. We use the same hyperparameters of Table III but set $N_{\text{epochs}} = 100$, with the aim of finding more reliable parameters. We repeat the optimization twice, adopting the same initial conditions but changing the device. We use the aforementioned qw5q and a different five-qubit chip constructed by IQM [71] and controlled using Zurich [72] Instruments (see Appendix C). We refer to this device as iqm5q. If the parameters obtained through RTQEM procedure are noise independent, we expect them to be generally valid. Namely, the optimal parameters obtained for one device should lead to a valid fit when deployed to a different one. This is illustrated in Fig. 8, where we report the results obtained by training individually on qw5q and on iqm5q with the same initial conditions, and then deploying the two sets of

TABLE III. Hyperparameters of the gradient descent on qw5q.

Param	N_{epochs}	N_{shots}	N_{train}	N_{params}	η	NumPy seed
Value	50	500	15	16	0.1	1234

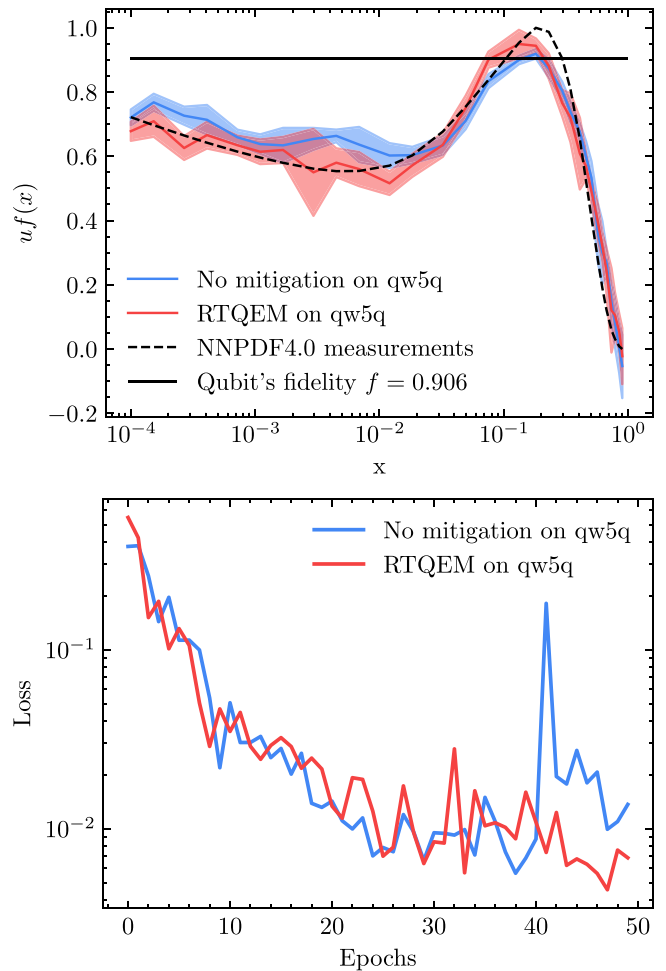


FIG. 7. Top: Estimates of $N_{\text{target}} = 30$ values of the u -quark PDF obtained by training the best qubit of the qw5q chip. The target values (black dashed line) are compared with our predictions after an unmitigated training (blue line) and a training with RTQEM (red line). The estimations are computed averaging over $N_{\text{runs}} = 10$ predictions for each x with the trained θ_{best} . The same prediction sets allow to calculate the standard deviations of the estimates, which are then used to draw the confidence intervals. Bottom: Loss function history as a function of the optimization epochs. The effective depolarizing parameter is $\lambda_{\text{eff}} = 0.07 \pm 0.03$.

obtained parameters solely on qw5q. The plotted estimates are computed by averaging on $N_{\text{runs}} = 20$ repeated predictions.

Finally, to further verify that the obtained parameters are indeed noise independent, we deploy the model obtained by training on qw5q via RTQEM on an exact simulator (green line in Fig. 8).

We calculate the MSE value for each described experiment following Eq. (17). All the results are collected in Table IV and confirm that the RTQEM training leads to noise-independent modelization.

It is interesting to compare the results presented in this section with those obtained by tackling this identical problem and using the same quantum circuit, but without applying the RTQEM algorithm [73]. What we notice is that the fit becomes appreciable even in the region most subject to noise, which would otherwise be inaccessible.

TABLE IV. MSE values for the models trained on the hardware. The column “Training” indicates the device where the training took place, whereas the column “Predictions” specifies the device where the model is deployed for testing.

Training	Predictions	Config.	N_{epochs}	MSE
qw5q	qw5q	Noisy	50	0.0055
qw5q	qw5q	RTQEM	50	0.0042
qw5q	qw5q	RTQEM	100	0.0013
iqm5q	qw5q	RTQEM	100	0.0037
qw5q	sim	RTQEM	100	0.0016

V. CONCLUSION

In this paper, we introduced a real-time quantum error mitigation routine designed to enhance the training process of variational quantum algorithms. We employed the importance Clifford sampling method at each learning step to mitigate noise in both the gradients of the loss function and the predictions. The RTQEM algorithm effectively reduces loss corruption without exacerbating loss concentration, thereby guiding the optimizer toward lower local minima of the loss function. We evaluated the RTQEM procedure using superconducting qubits and found that it improved the fit’s consistency by surpassing the limitations imposed by the hardware’s noise.

Our results demonstrate that the proposed algorithm effectively trains VQC models in noisy environments. Specifically,

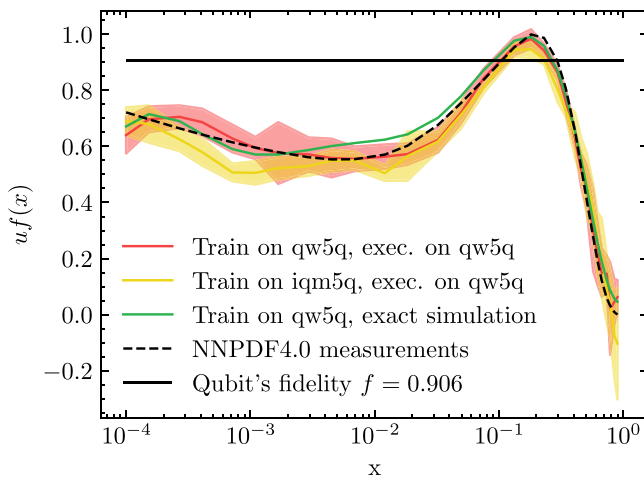


FIG. 8. Estimates of $N_{\text{target}} = 30$ values of the u -quark PDF obtained by training the better calibrated qubits of qw5q and iqm5q, respectively, with assignment fidelities $f_{\text{qw5q}} = 0.906$ and $f_{\text{iqm5q}} = 0.967$. The target values (black dashed line) are compared with our RTQEM predictions obtained by training for $N_{\text{epochs}} = 100$ on both qw5q (red line) and iqm5q (yellow line). We also show the predictions computed deploying in exact simulation mode the best parameters obtained through RTQEM training on qw5q (green line). The final average and standard deviation of the predictions are computed out of $N_{\text{runs}} = 20$ repetitions for each x using the parameters θ_{best} learned during training. In particular, the 1σ confidence intervals are shown in the plot. The effective depolarizing parameter is $\lambda_{\text{eff}} = 0.08 \pm 0.02$.

if the system’s noise remains constant or changes slowly, the noise map requires only a few updates during training, keeping the computational cost on par with the unmitigated training process.

While RTQEM is broadly applicable, its effectiveness naturally depends on factors such as the ansatz, the observable, and the underlying hardware noise. In the regimes considered here, the ansatz remains fully trainable, and RTQEM reliably mitigates cost-landscape corruption without inducing barren plateaus. More generally, the operational limits of the method are problem dependent, with performance decreasing near the boundaries where noise or cost concentration dominate.

Notably, by mitigating noise during training, we can derive parameters that closely approximate those of a noise-free environment. This adaptability allows us to deploy these parameters on a different device, even if it is subject to different noise. This capability paves the way for the potential integration of federated learning with quantum processors.

The extension of this approach to other QML pipelines that use expected values as predictors, as well as to other QEM methods, presents an intriguing avenue for future research. For instance, it can be applied to VQC models for supervised, unsupervised, and reinforcement learning scenarios in noisy environments.

ACKNOWLEDGMENTS

We would like to thank all members of the QRC laboratory for their support with the calibration of the devices. We thank Juan Cereijo, Andrea Pasquale, and Edoardo Pedicillo for the insightful discussions about the description of the devices used in this study. This project was supported by CERN’s Quantum Technology Initiative (QTI). M.R. was supported by CERN doctoral program. A.S. acknowledges financial support through the Spanish Ministry of Science and Innovation Grant No. SEV-2016-0597-19-4, the Spanish MINECO Grant No. PID2021-127726NB-I00, the Centro de Excelencia Severo Ochoa Program (SEV-2016-0597), and the CSIC Research Platform on Quantum Technologies (PTI-001). A.P. was supported by an Australian Government Research Training Program International Scholarship. S.C. thanks the CERN’s Department of Theoretical Physics hospitality during the elaboration of this manuscript.

DATA AVAILABILITY

The data that support the findings of this article are openly available [66].

APPENDIX A: GRADIENTS’ EVOLUTION

During the VQC training, the noisy gradients are of the same magnitude as the exact ones, indicating that we are in a regime where exponential concentration is not severe, as shown in Fig. 9.

APPENDIX B: NATIVE GATES

The native gates of the QRC superconducting quantum processors are $RX(\pm\pi/2)$, $RZ(\theta)$, and CZ gates [70]. They constitute a universal quantum gate set. These gates are com-

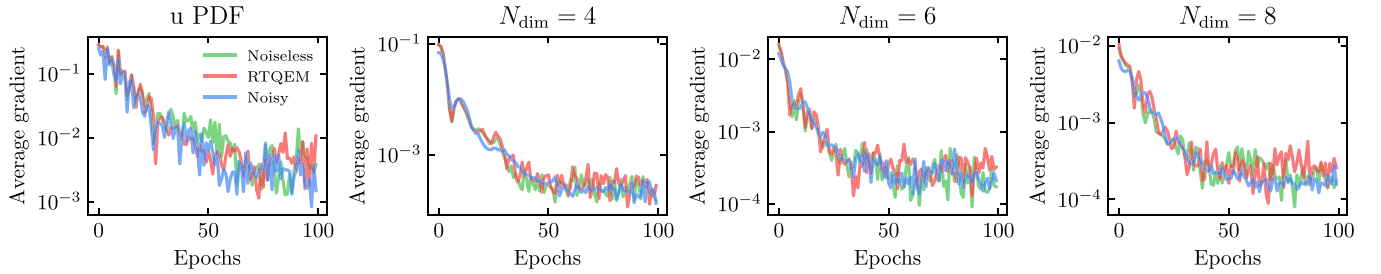


FIG. 9. Average gradients as a function of the optimization epochs. A noiseless simulation (green lines) is compared with unmitigated noisy simulation (blue lines) and RTQEM (red lines) for $N_{\text{dim}} = 4, 6, 8$ from the left to the right plots.

piled into microwave pulses following a specific set of rules [32]. For a circuit to be executable on hardware, it needs to be decomposed into these native gates. For instance, a general single-qubit unitary beaks into a sequence of five native gates,

$$U(\theta, \phi, \lambda) = RZ(\phi)RX(-\pi/2)RZ(\theta)RX(\pi/2)RZ(\lambda). \quad (\text{B1})$$

APPENDIX C: QUBITS' PARAMETERS

Relevant parameters of the qubits utilized in this study are presented in Table V, including

- (1) the qubit transition frequency $f_{01} = \omega_q/2\pi$ from $|0\rangle$ to $|1\rangle$;

- (2) the bare resonator frequency $f_{\text{res}} = \omega_{\text{res}}/2\pi$;
- (3) the readout frequency f_{read} (coupled resonator frequency);
- (4) the energy relaxation time T_1 ;
- (5) the dephasing time T_2 ;
- (6) the time τ_g required to execute a single RX gate.

In the same table, we also show the assignment fidelity $f = 1 - [P(1|0) - P(0|1)]/2$, where $P(i|j)$ is a misclassification metric, counting the states prepared as $|j\rangle$ but measured as $|i\rangle$. This value is primarily due to the calibration status of the devices, rather than construction limitations.

TABLE V. Parameters of the iqm5q and qw5q qubits employed in this study.

Qubit	f_{01} (GHz)	f_{res} (GHz)	f_{read} (GHz)	T_1 (μ s)	T_2 (μ s)	τ_g (ns)	f
qubit 4, iqm5q	4.0978	5.5047	5.5150	9.891	3.700	40	0.967
qubit 3, qw5q	6.7599	7.8000	7.8028	2.776	1.139	40	0.906

- [1] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
- [2] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Noisy intermediate-scale quantum algorithms, *Rev. Mod. Phys.* **94**, 015004 (2022).
- [3] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, *Nat. Commun.* **5**, 4213 (2014).
- [4] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, *New J. Phys.* **18**, 023023 (2016).
- [5] B. Bauer, D. Wecker, A. J. Millis, M. B. Hastings, and M. Troyer, Hybrid quantum-classical approach to correlated materials, *Phys. Rev. X* **6**, 031045 (2016).
- [6] T. Jones, S. Endo, S. McArdle, X. Yuan, and S. C. Benjamin, Variational quantum algorithms for discovering Hamiltonian spectra, *Phys. Rev. A* **99**, 062304 (2019).
- [7] E. Farhi and A. W. Harrow, Quantum supremacy through the quantum approximate optimization algorithm, *arXiv:1602.07674*
- [8] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, Noise-induced barren plateaus in variational quantum algorithms, *Nat. Commun.* **12**, 6961 (2021).
- [9] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, *Nat. Commun.* **9**, 4812 (2018).
- [10] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, *Nat. Commun.* **12**, 1791 (2021).
- [11] A. Arrasmith, M. Cerezo, P. Czarnik, L. Cincio, and P. J. Coles, Effect of barren plateaus on gradient-free optimization, *Quantum* **5**, 558 (2021).
- [12] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, Connecting ansatz expressibility to gradient magnitudes and barren plateaus, *PRX Quantum* **3**, 010313 (2022).
- [13] M. Ragone, B. N. Bakalov, F. Sauvage, A. F. Kemper, C. Ortiz Marrero, M. Larocca, and M. Cerezo, A Lie algebraic theory of barren plateaus for deep parameterized quantum circuits, *Nat. Commun.* **15**, 7172 (2024).
- [14] N. L. Diaz, D. García-Martín, S. Kazi, M. Larocca, and M. Cerezo, Showcasing a barren plateau theory beyond the dynamical Lie algebra, *arXiv:2310.11505*.

- [15] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, Error mitigation extends the computational reach of a noisy quantum processor, *Nature (London)* **567**, 491 (2019).
- [16] S. Wang, P. Czarnik, A. Arrasmith, M. Cerezo, L. Cincio, and P. J. Coles, Can error mitigation improve trainability of noisy variational quantum algorithms? *Quantum* **8**, 1287 (2024).
- [17] D. Qin, Y. Chen, and Y. Li, Error statistics and scalability of quantum error mitigation formulas, *npj Quantum Inf.* **9**, 35 (2023).
- [18] A. Strikis, D. Qin, Y. Chen, S. C. Benjamin, and Y. Li, Learning-based quantum error mitigation, *PRX Quantum* **2**, 040330 (2021).
- [19] M. Urbanek, B. Nachman, V. R. Pascuzzi, A. He, C. W. Bauer, and W. A. De Jong, Mitigating depolarizing noise on quantum computers with noise-estimation circuits, *Phys. Rev. Lett.* **127**, 270502 (2021).
- [20] S. A. Rahman, R. Lewis, E. Mendicelli, and S. Powell, Self-mitigating Trotter circuits for SU(2) lattice gauge theory on a quantum computer, *Phys. Rev. D* **106**, 074502 (2022).
- [21] R. C. Farrell, I. A. Chernyshev, S. J. M. Powell, N. A. Zemlevskiy, M. Illa, and M. J. Savage, Preparations for quantum simulations of quantum chromodynamics in $1 + 1$ dimensions. I. Axial gauge, *Phys. Rev. D* **107**, 054512 (2023).
- [22] A. N. Ciavarella, Quantum simulation of lattice QCD with improved Hamiltonians, *Phys. Rev. D* **108**, 094513 (2023).
- [23] R. C. Farrell, M. Illa, A. N. Ciavarella, and M. J. Savage, Scalable circuits for preparing ground states on digital quantum computers: The Schwinger model vacuum on 100 qubits, *PRX Quantum* **5**, 020315 (2024).
- [24] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Data re-uploading for a universal quantum classifier, *Quantum* **4**, 226 (2020).
- [25] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, *arXiv:1412.6980*.
- [26] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, *Phys. Rev. A* **98**, 032309 (2018).
- [27] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, *Phys. Rev. A* **99**, 032331 (2019).
- [28] S. Efthymiou, S. Ramos-Calderer, C. Bravo-Prieto, A. Pérez-Salinas, D. García-Martín, A. García-Saez, J. I. Latorre, and S. Carrazza, Qibo: A framework for quantum simulation with hardware acceleration, *Quantum Sci. Technol.* **7**, 015018 (2022).
- [29] S. Efthymiou, M. Lazzarin, A. Pasquale, and S. Carrazza, Quantum simulation with just-in-time compilation, *Quantum* **6**, 814 (2022).
- [30] S. Carrazza, S. Efthymiou, M. Lazzarin, and A. Pasquale, An open-source modular framework for quantum computing, *J. Phys.: Conf. Ser.* **2438**, 012148 (2023).
- [31] S. Efthymiou, *et al.*, qiboteam/qibo: Qibo 0.1.12 (2023), <https://doi.org/10.5281/zenodo.17822128>.
- [32] S. Efthymiou, A. Orgaz-Fuertes, R. Carobene, J. Cereijo, A. Pasquale, S. Ramos-Calderer, S. Bordoni, D. Fuentes-Ruiz, A. Candido, E. Pedicillo, M. Robbiati, Y. P. Tan, J. Wilkens, I. Roth, J. I. Latorre, and S. Carrazza, Qibolab: An open-source hybrid quantum operating system, *Quantum* **8**, 1247 (2024).
- [33] S. Efthymiou, *et al.*, qiboteam/qibolab: Qibolab 0.0.2 (2023), <https://doi.org/10.5281/zenodo.17814902>.
- [34] R. Carobene, A. Candido, J. Serrano, A. Orgaz-Fuertes, A. Giachero, and S. Carrazza, Qibosoq: An open-source framework for quantum circuit RFSoc programming, *Quantum Sci. Technol.* **10**, 035010 (2025).
- [35] A. Pasquale, S. Efthymiou, S. Ramos-Calderer, J. Wilkens, I. Roth, and S. Carrazza, Towards an open-source framework to perform quantum calibration and characterization, *arXiv:2303.10397*.
- [36] A. Pasquale, *et al.*, qiboteam/qibocal: Qibocal 0.0.1 (2023), <https://doi.org/10.5281/zenodo.18230323>.
- [37] E. Pedicillo, A. Pasquale, and S. Carrazza, Benchmarking machine learning models for quantum state classification *EPJ Web Conf.* **295**, 12007 (2024).
- [38] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature (London)* **549**, 195 (2017).
- [39] M. Schuld, I. Sinayskiy, and F. Petruccione, An introduction to quantum machine learning, *Contemp. Phys.* **56**, 172 (2014).
- [40] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, Variational quantum circuits for deep reinforcement learning *IEEE Access* **8**, 141007 (2020).
- [41] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, Quantum embeddings for machine learning, *arXiv:2001.03622*.
- [42] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, *Nature (London)* **567**, 209 (2019).
- [43] M. Incudini, D. L. Bosco, F. Martini, M. Grossi, G. Serra, and A. D. Pierro, Automatic and Effective Discovery of Quantum Kernels, in *IEEE Transactions on Emerging Topics in Computational Intelligence* (IEEE, 2024).
- [44] A. Pérez-Salinas, J. Cruz-Martinez, A. A. Alhajri, and S. Carrazza, Determining the proton content with a quantum computer, *Phys. Rev. D* **103**, 034027 (2021).
- [45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, *Nature (London)* **323**, 533 (1986).
- [46] J. Duchi, E. Hazan, and Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *J. Mach. Learn. Res.* **12**, 2121 (2011).
- [47] S. Ruder, An overview of gradient descent optimization algorithms, *arXiv:1609.04747*.
- [48] J. Schmidhuber, Deep learning in neural networks: An overview, *Neural. Net.* **61**, 85 (2015).
- [49] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, The power of quantum neural networks, *Nat. Comput. Sci.* **1**, 403 (2021).
- [50] G. E. Crooks, Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition, *arXiv:1905.13311*.
- [51] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, General parameter-shift rules for quantum gradients, *Quantum* **6**, 677 (2022).
- [52] A. Mari, T. R. Bromley, and N. Killoran, Estimating the gradient and higher-order derivatives on quantum hardware, *Phys. Rev. A* **103**, 012405 (2021).
- [53] L. Banchi and G. E. Crooks, Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule, *Quantum* **5**, 386 (2021).

- [54] R. Sweke, F. Wilde, J. Meyer, M. Schuld, P. K. Faehrmann, B. Meynard-Piganeau, and J. Eisert, Stochastic gradient descent for hybrid quantum-classical optimization, *Quantum* **4**, 314 (2020).
- [55] M. Robbiati, S. Efthymiou, A. Pasquale, and S. Carrazza, A quantum analytical Adam descent through parameter shift rule using Qibo, *PoS ICHEP2022*, 206 (2022).
- [56] Y. Li and S. C. Benjamin, Efficient Variational Quantum Simulator Incorporating Active Error Minimization, *Phys. Rev. X* **7**, 021050 (2017).
- [57] K. Temme, S. Bravyi, and J. M. Gambetta, Error mitigation for short-depth quantum circuits, *Phys. Rev. Lett.* **119**, 180509 (2017).
- [58] A. Lowe, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, Unified approach to data-driven quantum error mitigation, *Phys. Rev. Res.* **3**, 033098 (2021).
- [59] P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, Error mitigation with Clifford quantum-circuit data, *Quantum* **5**, 592 (2021).
- [60] A. Sopena, M. H. Gordon, G. Sierra, and E. López, Simulating quench dynamics on a digital quantum computer with data-driven error mitigation, *Quantum Sci. Technol.* **6**, 045003 (2021).
- [61] E. van den Berg, Z. K. Mineev, and K. Temme, Model-free readout-error mitigation for quantum expectation values, *Phys. Rev. A* **105**, 032620 (2022).
- [62] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, and T. E. O'Brien, Quantum error mitigation, *Rev. Mod. Phys.* **95**, 045005 (2023).
- [63] S. Aaronson and D. Gottesman, Improved simulation of stabilizer circuits, *Phys. Rev. A* **70**, 052328 (2004).
- [64] H. Pashayan, O. Reardon-Smith, K. Korzekwa, and S. D. Bartlett, Fast Estimation of Outcome Probabilities for Quantum Circuits, *PRX Quantum* **3**, 020361 (2022).
- [65] B. Nachman, M. Urbanek, W. A. De Jong, and C. W. Bauer, Unfolding quantum computer readout noise, *npj Quantum Inf.* **6**, 84 (2020).
- [66] M. Robbiati, A. Sopena, BrunoLiegiBastonLiegi, and S. Carrazza, qiboteam/rtqem: Version 0.0.1 (2023), <https://doi.org/10.5281/zenodo.10092274>.
- [67] R. D. Ball, S. Carrazza, J. Cruz-Martinez, L. D. Debbio, S. Forte, T. Giani, S. Iranipour, Z. Kassabov, J. I. Latorre, E. R. Nocera, R. L. Pearson, J. Rojo, R. Stegeman, C. Schwan, M. Ubiali, C. Voisey, and M. Wilson, The path to proton structure at 1% accuracy, *Eur. Phys. J. C* **82**, 428 (2022).
- [68] QuantWare (2023), <https://quantware.com>.
- [69] Qblox Cluster (2023), https://docs.qblox.com/en/main/products/architecture/cluster/rf_cluster.html.
- [70] qibolab_platforms_qrc (2023), https://github.com/qiboteam/qibolab_platforms_qrc.
- [71] IQM Quantum Computers (2023), <https://meetiqm.com>.
- [72] Zurich Instruments Quantum Computing Control System (2023), <https://www.zhinst.com>.
- [73] A. D'Elia, *et al.*, Characterization of a transmon qubit in a 3D cavity for quantum machine learning and photon counting, *Appl. Sci.* **14**, 1478 (2024).