Real-time error mitigation for variational optimization on quantum hardware

Matteo Robbiati,[1,2] Alejandro Sopena,[3] Andrea Papaluca,[4,5] and Stefano Carrazza[1,2,5]

[1] *TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano and INFN Sezione di Milano, Milan, Italy.*
[2] *CERN, Theoretical Physics Department, CH-1211 Geneva 23, Switzerland.*
[3] *Instituto de Física Teórica, UAM-CSIC, Universidad Autónoma de Madrid, Cantoblanco, Madrid, Spain*
[4] *School of Computing, The Australian National University, Canberra, ACT, Australia*
[5] *Quantum Research Centre, Technology Innovation Institute, Abu Dhabi, UAE.*

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 coadiuvate the task of fitting functions on quantum chips with VQCs. While state-of-the-art QEM methods cannot adress 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 (PDF) on a superconducting single-qubit device, and we further analyzed the scalability of the proposed technique by simulating a multidimensional fit with up to 8 qubits.

In the era of Noisy Intermediate Scale Quantum (NISQ) [1, 2] devices, Variational Quantum Algorithms (VQA) 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 ansatze have been proposed, such as the 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 an usable amount of logical qubits, or exploit the available NISQ hardware by cleaning its results. 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.

In this work, we use in particular the Importance Clifford Sampling (ISC) [17], which implements 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 which, once learned, can be used to clean the noisy results. 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 multi-dimensional regression problems. Firstly, we train a VQC to tackle a particularly interesting High Energy Physics problem: determining the Parton Distribution Function (PDF) of the the u-quark, one of the proton contents. In a second step, we define a multi-dimensional target to study the impact of the RTQEM procedure when the VQA involves an increasing number of qubits.

Data re-uploading [19] is used to encode data into the the model, while we implement an hardware-compatible Adam [20] optimizer for the training. We calculate gradients with respect to the variational parameters using the Parameter Shift Rule [21, 22] (PSR). This optimisation 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 allow for training 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 Centre (QRC) of the Technology Innovation Institute (TII).

The whole work has been realized using the Qibo framework, which offers `Qibo` [23–26] as high-level language API to write quantum computing algorithms, `Qibolab` [27–29] as quantum control tool and `Qibocal` [30, 31] to perform quantum characterization and calibration routines.

The outline is as follows. In Section 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 Section II, we

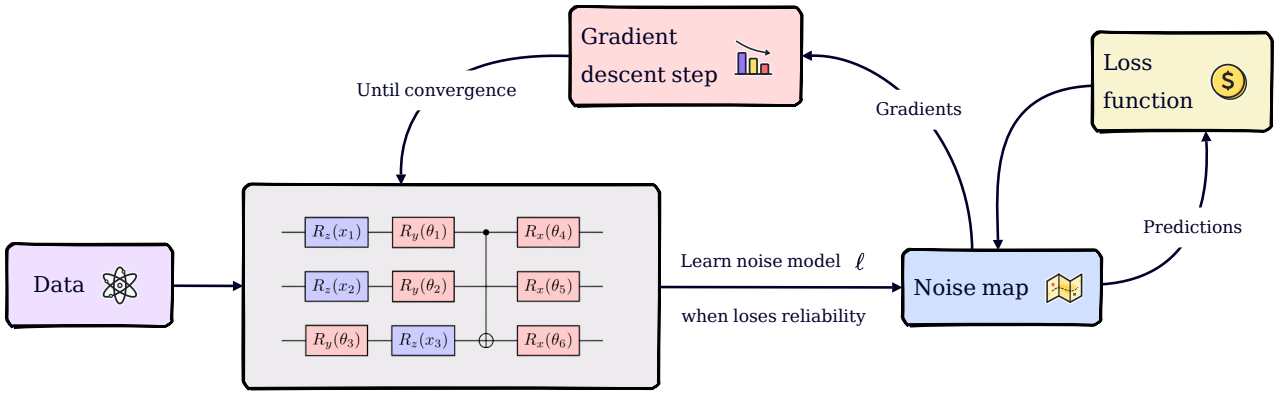

FIG. 1. RTQEM pipeline.

present the effects of noise during the training process and in Section II A an outline of the error mitigation method we experimented with to alleviate them. Finally, we report the results of our experiments both with noisy simulations and real superconducting qubits deployment in Section 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 (QML) arises when using Quantum Computing (QC) tools to tackle ML problems [21, 32, 33].

In the classical scenario, given an n-dimensional input variable $\boldsymbol{x}$, a parametric model is requested to estimate a target variable y, which is related to $\boldsymbol{x} = (x_1, \ldots, x_n)$ through some hidden law $y = g(\boldsymbol{x})$. The model estimations y_{est} are then compared with some measured groundtruth 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. The variational parameters $\boldsymbol{\theta}$ 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 Variational Quantum Circuits (VQC) [34], 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 $|\psi_i\rangle$ 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 $\hat{B}$,

$$f(\boldsymbol{\theta})_{\hat{B}} = \langle\psi_i|\mathcal{U}^\dagger(\boldsymbol{\theta})\hat{B}\,\mathcal{U}(\boldsymbol{\theta})|\psi_i\rangle \ . \tag{1}$$

In what follows we remove the dependence on $\hat{B}$ of the symbol f for simplicity. Various methods exist to embed input data into a QML process [35–37]; in this work, we employ the re-uploading strategy [19]. 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 [19] method is built by defining a parameterized layer made of fundamental uploading gates which accepts the input data $\boldsymbol{x}$ to be uploaded. Then, the re-uploading of the variable is achieved by building a circuit composed of a sequence of uploading layers. Inspired by [38], we build our ansatz by defining the following fundamental uploading gate,

$$L(x_j|\boldsymbol{\theta}_{l,j}) = R_z(\theta_3\, x_j + \theta_4) R_y(\theta_1\, \kappa(x_j) + \theta_2) \ , \tag{2}$$

where x_j is the j-th component of the input data and with $\boldsymbol{\theta}_{l,j}$ we denote the four-parameters vector composing the gate which uploads x_j at the ansatz layer l. The information x_j is uploaded twice in each L, first in the R_z and second in the R_y through an activation function $\kappa(x_j)$. To embed the n components of $\boldsymbol{x}$ into the ansatz, we build a n-qubit circuit based on the Hardware Efficient Ansatz, where the single-qubit gates are the fundamental uploading gates, and entanglement is generated with CNOT gates, as shown in Fig. 2.

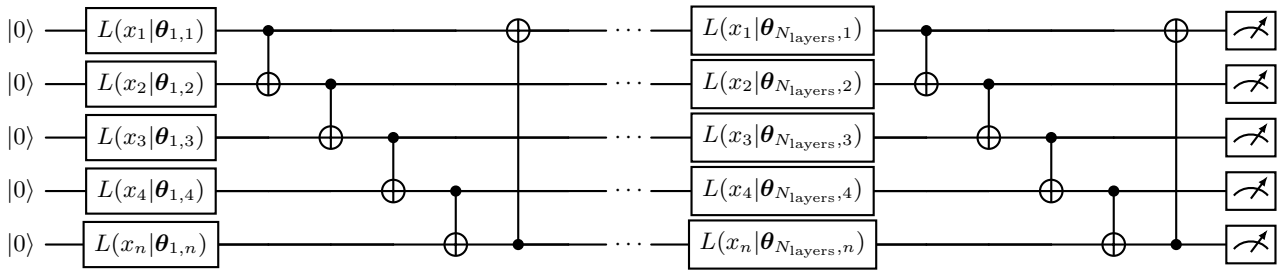

FIG. 2. Circuit ansatz to reupload the data $\boldsymbol{x}$ with n qubits and N_{layers} layers.

C. Gradient descent on hardware

Gradient-based optimizers [20, 39–41] are commonly employed in machine learning problems, particularly when using Neural Networks [42] (NNs) as models. In the QML context, VQCs are utilized to construct Quantum Neural Networks [43], 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 a 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 [39] 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 [22, 44–47] (PSR), which allows for exactly evaluating quantum gradients directly on the hardware [22]. Given f as introduced in (1) and considering a single parameter $\mu \in \boldsymbol{\theta}$ affecting a single gate whose hermitian generator has at most two eigenvalues, the PSR allows for the calculation

$$\partial_\mu f = r\big(f(\mu^+) - f(\mu^-)\big) \ , \tag{3}$$

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 $\mathcal{U}(\boldsymbol{\theta})$ 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 $s = \pi/2$ [21].

2. *Evaluating gradients of a re-uploading 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,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla J(\boldsymbol{\theta}_i) \ , \tag{4}$$

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 (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 circuit execution of the algorithm. We followed the same optimization stategy described in [48, 49], defining a Mean-Squared Error loss function,

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

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

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 [20] optimizer stands out due to its robustness when dealing with complex parameters 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 ansatzes such as Hardware Efficient Ansatz (HEA), 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 (NIBP) [50]. 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,

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

where $-1 < q_X, q_Y, q_Z < 1$ and σ denotes the local Pauli operators $\{\sigma_x, \sigma_y, \sigma_z\}$. 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,

$$f_{\text{noisy}} = \text{Tr}\left[Z\left(\mathcal{M} \circ \mathcal{P} \circ L_{N_{\text{layers}}} \circ \cdots \circ \mathcal{P} \circ L_1 \circ \mathcal{P}\right)(|0\rangle\langle 0|)\right] . \tag{7}$$

The NIBP translates into a concentration of the expectation value around 0 [50],

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

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 non-resistant loss functions.

A. Error Mitigation

Recent research [51–56] 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, impliying 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 [57], setting up a trade-off between the improvement due to bias reduction and the worsening caused by increased variance.

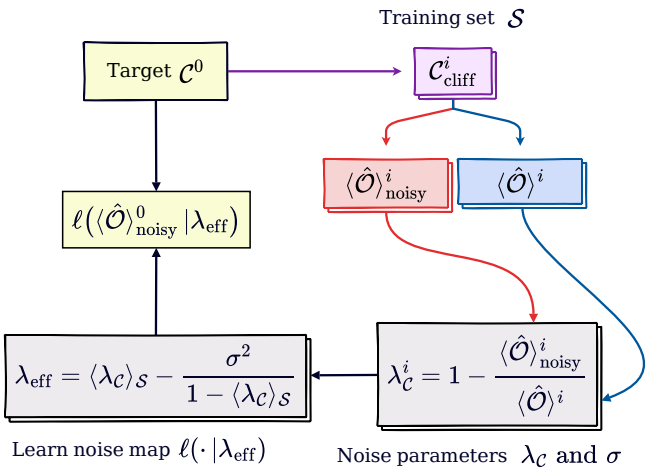

FIG. 3. ICS error mitigation.

Most methods have a negative impact on resolvability, but linear ansatz methods show a neutral effect. Among all of them, Importance Clifford Sampling [17] stands out for its scalability and resource cost.

1. *Importance Clifford Sampling (ICS)*

Suppose we want to estimate the expected value of an observable $\hat{\mathcal{O}}$ 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 $\langle\hat{\mathcal{O}}\rangle^0_{\text{noisy}}$ different from the true one $\langle\hat{\mathcal{O}}\rangle^0$. The idea behind Importance Clifford Sampling (ICS) is to generate a set of training Clifford circuits $\mathcal{S} = \{\mathcal{C}^i\}_{i=1}^n$ with the same circuit frame as the original one $\mathcal{C}^0$. The classical computation of noiseless expected values of Clifford circuits is efficient [58, 59]. This enables us to compute the ideal expected values $\{\langle\hat{\mathcal{O}}\rangle^i\}_{i=1}^n$ through simulation, as well as the noisy expected values $\{\langle\hat{\mathcal{O}}\rangle^i_{\text{noisy}}\}_{i=1}^n$ when evaluating them on hardware.

When $\hat{\mathcal{O}}$ is a Pauli string, the noise-free expected values will concentrate on -1, 0, 1 [58]. Furthermore, as discussed in [17], not all the clifford circuits are error sentive. In particular, we only need circuits whose expected values on Pauli's are ± 1. We refer to these circuits as *non-zero* circuits for simplicity. Unfortunately, sampling *non-zero* 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 *non-zero* circuits are built by adding an additional 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 $\langle\hat{\mathcal{O}}\rangle_{\text{noisy}}$ and $\langle\hat{\mathcal{O}}\rangle$. The structure of the model ℓ can be inspired by considering the action of a global depolarising channel with depolarising

parameter λ,

$$\langle\hat{\mathcal{O}}\rangle_{\text{noisy}} = (1-\lambda)\langle\hat{\mathcal{O}}\rangle + \frac{\lambda}{d}\,\text{Tr}(\hat{\mathcal{O}})\ , \tag{9}$$

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,

$$\langle\hat{\mathcal{O}}\rangle^i_{\text{noisy}} = (1-\lambda^i_{\mathcal{C}})\langle\hat{\mathcal{O}}\rangle^i\ , \tag{10}$$

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

$$\lambda_{\text{eff}} = \lambda_0 - \frac{\sigma^2}{1-\lambda_0}\ . \tag{11}$$

This translates into the noise map,

$$\ell\big(\langle\hat{O}\rangle|\lambda_{\text{eff}}\big) = \frac{(1-\lambda_0)}{(1-\lambda_0)^2+\sigma^2}\langle\hat{O}\rangle_{\text{noisy}}\ . \tag{12}$$

The average depolarising 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 (12) effectively handles symmetric readout noise, but fails with asymmetric noise. For these situations, we employ Bayesian iterative unfolding [60] to mitigate measurement errors in advance.

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

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

$$D(z, \ell(z)) = |z - \ell(z)| \tag{13}$$

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 *non-zero* test circuit to maximize the bias. If an arbitrary set threshold value ε_ℓ is exceeded, the noise map is relearnt from scratch. A schematic representation of the proposed procedure is reported in Alg. 1.

Algorithm 1: RTQEM

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

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

IV. VALIDATION

We propose two different experiments to test the RTQEM algorithm introduced above. Firstly, 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 real superconducting single-qubit chip. The programs to reproduce such simulations can be found at [61].

A. Simulation

In this section, we benchmark different levels of error mitigation by conducting both noisy and noiseless classical simulations with $N_{\text{shots}} = 10000$ shots as outlined in Tab. I. The VQC shown in Fig. 2 is used as ansatz and the noise is described by the noise model presented in Section II. We first consider a static-noise scenario in Section IV A 1, while in Section IV A 2 we let the noise vary over time.

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

TABLE I. Mean-Squared Error Values quantifying the distance between the target NNPDF4.0 measurements and the estimates obtained averaging on N_{runs} prediction sets.

1. *Static-noise scenario*

The following simulations are performed using a static local Pauli noise model where we set the following noise

parameters $q_M = 0.005$, $q_X = 0.007$, $q_Y = 0.003$ and $q_Z = 0.002$.

We first consider a one-dimensional target, namely, the u-quark Parton Distribution Function (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 [62]. The obtained results are presented in Fig. 4, demonstrating how the RTQEM approach allows the training to converge to the right solution.

The noisy simulation has a clear limitation due to loss concentration 7, which prevents the predictions from surpassing the bound $y \simeq 0.85$. The same limit can be observed in Fig. 4, where the loss value is not able to fall below the threshold with the unmitigated training. If we try to correct the predictions only after the training (f-QEM), we see how the QEM makes the region above the bound accessible, but the fit does not improve. This is expected, as the noise shifts the position of the minima of the loss function making it impossible to recover the true minimum with just a final rescaling pass. 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 multi-dimensional function. In particular, we consider

$$f_{\text{ndim}}(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{i=1}^{N_{\text{dim}}} \left[\cos\left(\theta_i x_i\right)^i + (-1)^{i-1} \theta_i x_i\right], \qquad (14)$$

where both data $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$ have dimension N_{dim} and the index i runs over the problem dimensions. The target f_{ndim} is rescaled in order to occupy the range $[0, 1]$. We consider N_{data} values for each $x_i \in \boldsymbol{x}$ homogeneously distributed in the range $[0, 1]$. The ansatz is the N_{dim}-qubit circuit 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,

$$\text{MSE} = \frac{1}{N_{\text{data}}} \sum_{j=1}^{N_{\text{data}}} \left(\bar{y}_{j,\text{est}} - y_{j,\text{meas}}\right)^2 , \qquad (15)$$

where $\bar{y}_{j\text{est}}$ is the average estimate over N_{runs}. The MSE associated to each fit is shown in Tab. 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

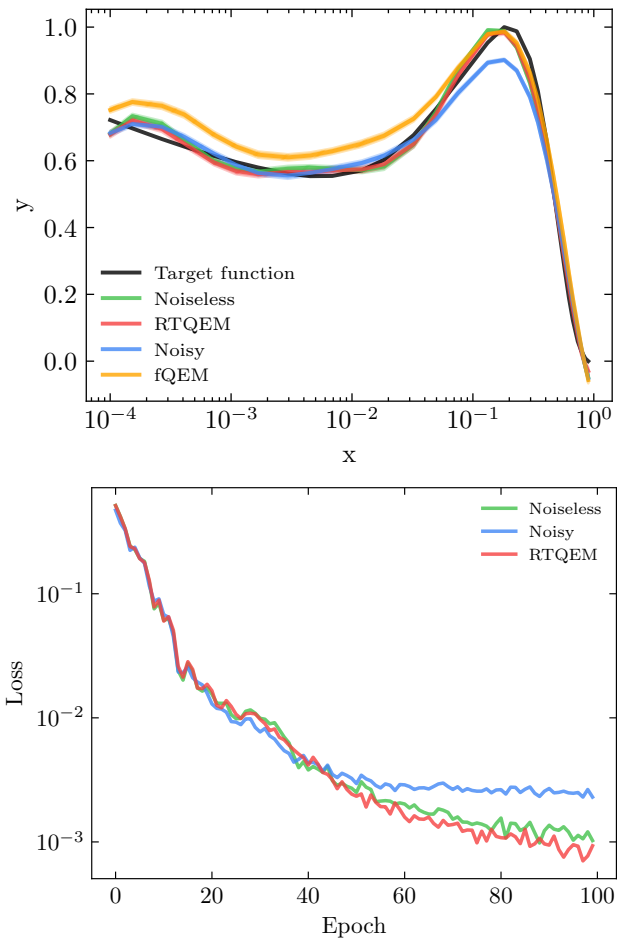

FIG. 4. Estimates of u-quark PDF associated to $N_{\text{data}} = 50$ momentum fraction values sampled logarithmically in $[0, 1]$. The NNPDF4.0 measures (black line) are compared with results obtained through noiseless simulation (green line), noisy simulation (blue line), noisy simulation with mitigation applied to the final predictions (yellow line) and real-time mitigated noisy simulation (red line). The effective depolarising 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.

Target	$\text{MSE}_{\text{noiseless}}$	$\text{MSE}_{\text{noisy}}$	MSE_{fqem}	$\text{MSE}_{\text{rtqem}}$
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

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

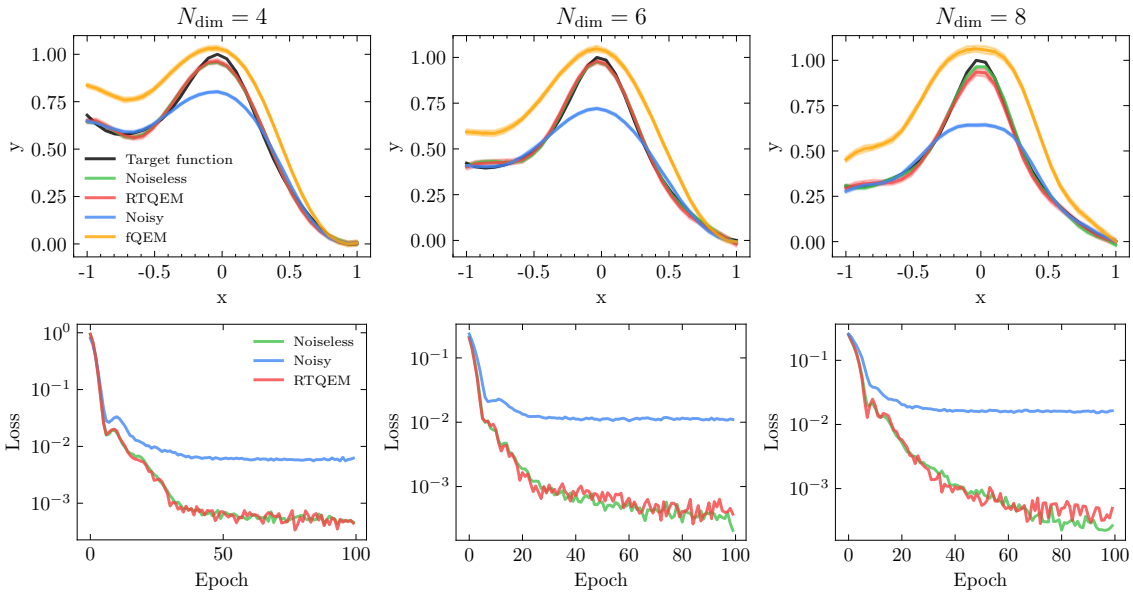

FIG. 5. Predictions for the multidimensional function $f_{\rm ndim}$ with $N_{\rm dim} = 4, 6, 8$ from left to right. The exact predictions (black line) are compared with results obtained through noiseless simulation (green line), noisy simulation (blue line), noisy simulation with mitigation applied to the final predictions (yellow line) and real-time mitigated noisy simulation (red line). The effective depolarising parameters $\lambda_{\rm eff}$ are 0.22 ± 0.02 ($N_{\rm dim} = 4$), 0.31 ± 0.03 ($N_{\rm dim} = 6$) and 0.41 ± 0.02 ($N_{\rm dim} = 8$). The final predictions are computed averaging on $N_{\rm runs} = 20$ predictions calculated for each of the $N_{\rm data} = 30$ 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.

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 the noise model in each epoch. In particular, the initial parameters vector $\boldsymbol{q}^0 = (q_X^0, q_Y^0, q_Z^0)$ is moved in its three-dimensional space following a procedure similar to a Random Walk (RW) on a lattice. Namely, the component q_j is evolved from epoch k to epoch $k+1$ as

$$q_j^{(k+1)} = q_j^k + r\delta, \tag{16}$$

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 $\mathrm{RW}^N_{\sigma_\delta}$. The readout noise parameter is kept fixed at the value of $q_M = 0.005$. In this evolving scenario, when the metric (13) exceeds a certain threshold ε_ℓ, the mitigation parameter $\lambda_{\rm eff}$ (11) is updated.

To evaluate the effect of relearning the noise on the training process, we track the evolution of the loss function 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

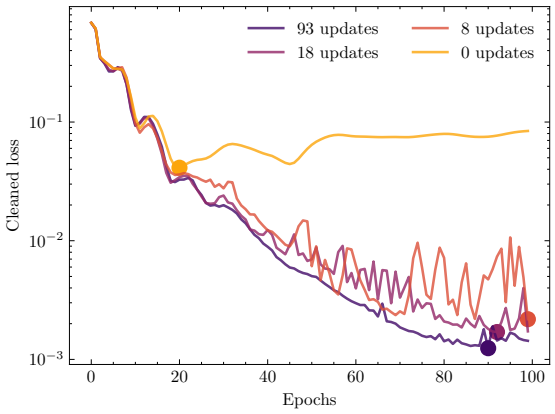

FIG. 6. Four RTQEM training simulations sharing the same initial conditions. The initial local Pauli parameters $\boldsymbol{q}^0 = (0.005, 0.005, 0.005)$ are evolved under a $\mathrm{RW}_{0.002}$. The readout noise parameter has been kept fixed to $q_M = 0.005$. The target function is the u-quark PDF, and $N_{\rm layers} = 4$ are used with $N_{\rm data} = 30$ 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, $\lambda_{\rm eff}$ is re-learned $u = \{93, 18, 8, 0\}$ times, respectively. We show the loss function values computed using the training parameters at each iterations but deployed in a noiseless scenario.

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 not updating the noise map and when updating it 94 times in a training of 100 epochs is $\mathcal{O}\left(10^{-3}\right)$. This suggests that a minor additional classical computational loss 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 Quantum Research Center (QRC) in the Technology Innovation Institute (TII). The high-level algorithm is implemented with `Qibo` [23–26] and then translated into pulses and executed on the hardware through the `Qibolab` [27, 28] framework (see Appendix B). The qubit calibration and characterization have been performed using `Qibocal` [30, 31]. In particular, we use one qubit of a five-qubits chip constructed by QuantWare [63] and controlled using Qblox [64] instruments. We refer to this device as `qw5q`.

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

An estimate to the bound imposed by the noise is provided, in this case, by the assignement fidelity of the used qubits, which are collected in dedicated runcards describing the current status of the QRC devices [65]. To avoid the u-quark PDF comfortably resting below this threshold 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 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. Furthermore, as the bound scales exponentially in the number of qubits [8], the number of shots needed will increase exponentially as well.

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

TABLE III. Hyper parameters of the gradient descent on `qw5q`

We perform a gradient descent on the better calibrated qubit of `qw5q` using the parameters collected in Tab. III.

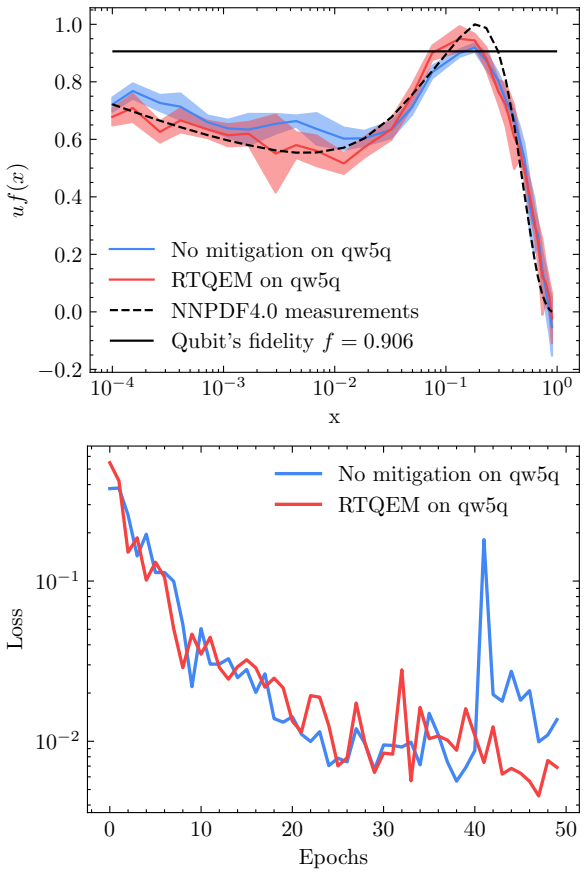

FIG. 7. Above, estimates of $N_{\text{target}} = 30$ values of the u-quark PDF obtained by training the best qubit of the `qw5q` chip. The target values (black dashed line) are compared with our predictions after an unmitigated training (blue line) and a training with RTQEM (red line). The estimations are computed averaging over $N_{\text{runs}} = 10$ predictions for each x with the trained $\boldsymbol{\theta}_{\text{best}}$. The same prediction sets allow to calculate the standard deviations of the estimates, which are then used to draw the 1σ confidence intervals. Below, loss function history as function of the optimization epochs. The effective depolarising parameter is $\lambda_{\text{eff}} = 0.07 \pm 0.03$.

The training has been performed for both the unmitigated and the RTQEM approaches. After training, we repeat $N_{\text{runs}} = 10$ times the predictions for each one of $N_{\text{target}} = 30$ target values of x sampled logarithmically 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 Tab. IV. This proves that the RTQEM procedure gives access to regions which are naturally forbidden to the raw training.

As a second test, we push forward the RTQEM training

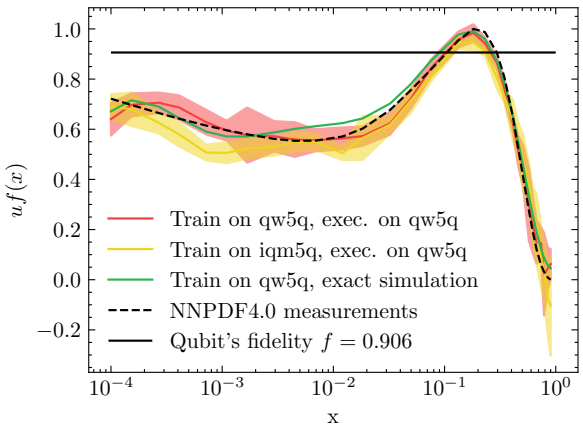

FIG. 8. Estimates of $N_{\text{target}} = 30$ values of the u-quark PDF obtained by training the better calibrated qubits of `qw5q` and `iqm5q`, respectively with assignment fidelities $f_{\text{qw5q}} = 0.906$ and $f_{\text{iqm5q}} = 0.967$. The target values (black dashed line) are compared with our 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 $N_{\text{runs}} = 20$ repetition for each x using the parameters $\boldsymbol{\theta}_{\text{best}}$ learnt during training. In particular, the 1σ confidence intervals are shown in the plot. The effective depolarising parameter is $\lambda_{\text{eff}} = 0.08 \pm 0.02$.

by performing a longer optimization. We use the same hyper-parameters of Tab. 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 [66] and controlled using Zurich [67] Instruments. 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 totally different one. This is illustrated in Fig. 8, where we report the results obtained by training individually `qw5q` and `iqm5q` with the same initial conditions, and then deploying the two sets of obtained parameters on `qw5q` only. 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 `qw5q` via RTQEM in an exact simulator (green line in Fig. 8).

We calculate the MSE value for each described experiment following 15. All the results are collected in Tab. IV, and confirm that the RTQEM training leads to noise-indipendent modelization.

V. CONCLUSION

In this work we presented a novel Real Time Quantum Error Mitigation routine to improve the training process of Variational Quantum Algorithms. The Importance Clifford Sampling method is used at each step of the learning process to clean from the noise both the gradients of the loss function and the predictions. The RTQEM algorithm allows for improved training by reducing loss corruption without worsening loss concentration, allowing the optimiser to follow the descent towards lower local minima of the loss function. We finally tested the RTQEM procedure on a real superconducting qubit, finding it to improve the consistency of the fit by overcoming the bounds imposed by the hardware's noise.

We proved that the proposed algorithm is helpful for training VQC models in the presence of noise. In particular, if the noise of the system remains steady or changes slowly enough, it is possible to train the noise map only a few times during training, making the computational cost comparable to the unmitigated training process. Moreover, by mitigating the noise during the training, it is possible to obtain parameters close to the noise-free parameters. This enables us to use them on another device that may be affected by a different noise.

The extension of this approach to other QML pipelines that make use of expected values as predictors, as well as to other QEM methods, represents an interesting research aveneu that we leave to a future work.

ACKNOWLEDGMENTS

We would like to thank everyone working in the QRC lab for supporting us with the devices' calibration.

This project is supported by CERN's Quantum Technology Initiative (QTI). MR is supported by CERN doctoral program. AS acknowledges financial support through the Spanish Ministry of Science and Innovation grant SEV-2016-0597-19-4, the Spanish MINECO grant PID2021- 127726NB-I00, the Centro de Excelencia Severo Ochoa Program SEV-2016-0597 and the CSIC Research Platform on Quantum Technologies PTI-001. AP was supported by an Australian Government Research Training Program International Scholarship. SC

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

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.

thanks the TH hospitality during the elaboration of this manuscript.

[1] J. Preskill, 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, Reviews of Modern Physics **94** (2022), 10.1103/revmodphys.94.015004.

[3] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, Nature Communications **5** (2014), 10.1038/ncomms5213.

[4] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, New Journal of Physics **18**, 023023 (2016).

[5] B. Bauer, D. Wecker, A. J. Millis, M. B. Hastings, and M. Troyer, Phys. Rev. X **6**, 031045 (2016).

[6] T. Jones, S. Endo, S. McArdle, X. Yuan, and S. C. Benjamin, Phys. Rev. A **99**, 062304 (2019).

[7] E. Farhi and A. W. Harrow, arXiv: Quantum Physics (2016).

[8] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, Nature Communications **12**, 6961 (2021).

[9] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Nature Communications **9**, 4812 (2018).

[10] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Nature Communications **12**, 1791 (2021).

[11] A. Arrasmith, M. Cerezo, P. Czarnik, L. Cincio, and P. J. Coles, Quantum **5**, 558 (2021).

[12] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, PRX Quantum **3**, 010313 (2022).

[13] M. Ragone, B. N. Bakalov, F. Sauvage, A. F. Kemper, C. O. Marrero, M. Larocca, and M. Cerezo, (2023), 10.48550/arXiv.2309.09342.

[14] N. L. Diaz, D. García-Martín, S. Kazi, M. Larocca, and M. Cerezo, (2023), 10.48550/arXiv.2310.11505.

[15] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, Nature **567**, 491 (2019).

[16] S. Wang, P. Czarnik, A. Arrasmith, M. Cerezo, L. Cincio, and P. J. Coles, (2021), 10.48550/arXiv.2109.01051.

[17] D. Qin, Y. Chen, and Y. Li, npj Quantum Information **9** (2023), 10.1038/s41534-023-00707-7.

[18] A. Strikis, D. Qin, Y. Chen, S. C. Benjamin, and Y. Li, PRX Quantum **2**, 040330 (2021).

[19] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Quantum **4**, 226 (2020).

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," (2017), arXiv:1412.6980 [cs.LG].

[21] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Physical Review A **98** (2018), 10.1103/physreva.98.032309.

[22] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Physical Review A **99** (2019), 10.1103/physreva.99.032331.

[23] S. Efthymiou, S. Ramos-Calderer, C. Bravo-Prieto, A. Pérez-Salinas, D. García-Martín, A. Garcia-Saez, J. I. Latorre, and S. Carrazza, Quantum Science and Technology **7**, 015018 (2021).

[24] S. Efthymiou, M. Lazzarin, A. Pasquale, and S. Carrazza, Quantum **6**, 814 (2022).

[25] S. Carrazza, S. Efthymiou, M. Lazzarin, and A. Pasquale, Journal of Physics: Conference Series **2438**, 012148 (2023).

[26] S. Efthymiou *et al.*, "qiboteam/qibo: Qibo 0.1.12," (2023).

[27] 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," (2023), arXiv:2308.06313 [quant-ph].

[28] S. Efthymiou *et al.*, "qiboteam/qibolab: Qibolab 0.0.2," (2023).

[29] R. Carobene, A. Candido, J. Serrano, A. Orgaz-Fuertes, A. Giachero, and S. Carrazza, "Qibosoq: an open-source framework for quantum circuit rfsoc programming," (2023), arXiv:2310.05851 [quant-ph].

[30] 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," (2023), arXiv:2303.10397 [quant-ph].

[31] A. Pasquale *et al.*, "qiboteam/qibocal: Qibocal 0.0.1," (2023).

[32] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature **549**, 195 (2017).

[33] M. Schuld, I. Sinayskiy, and F. Petruccione, Contemporary Physics **56**, 172 (2014).

[34] 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," (2020), arXiv:1907.00397 [cs.LG].

[35] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, "Quantum embeddings for machine learning," (2020), arXiv:2001.03622 [quant-ph].

[36] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Nature **567**, 209 (2019).

[37] M. Incudini, F. Martini, and A. D. Pierro, "Structure learning of quantum embeddings," (2022), arXiv:2209.11144 [quant-ph].

[38] A. Pérez-Salinas, J. Cruz-Martinez, A. A. Alhajri, and S. Carrazza, Phys. Rev. D **103**, 034027 (2021).

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Nature **323**, 533 (1986).

[40] J. Duchi, E. Hazan, and Y. Singer, Journal of Machine Learning Research **12**, 2121 (2011).

[41] S. Ruder, "An overview of gradient descent optimization algorithms," (2017), arXiv:1609.04747 [cs.LG].

[42] J. Schmidhuber, Neural Networks **61**, 85 (2015).

[43] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, Nature Computational Science **1**, 403 (2021).

[44] G. E. Crooks, "Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition," (2019), arXiv:1905.13311 [quant-ph].

[45] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, Quantum **6**, 677 (2022).

[46] A. Mari, T. R. Bromley, and N. Killoran, Physical Review A **103** (2021), 10.1103/physreva.103.012405.

[47] L. Banchi and G. E. Crooks, Quantum **5**, 386 (2021).
[48] R. Sweke, F. Wilde, J. Meyer, M. Schuld, P. K. Faehrmann, B. Meynard-Piganeau, and J. Eisert, Quantum **4**, 314 (2020).
[49] M. Robbiati, S. Efthymiou, A. Pasquale, and S. Carrazza, "A quantum analytical adam descent through parameter shift rule using qibo," (2022), arXiv:2210.10787 [quant-ph].
[50] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, Nature Communications **12**, 6961 (2021).
[51] Y. Li and S. C. Benjamin, Physical Review X **7**, 021050 (2017).
[52] K. Temme, S. Bravyi, and J. M. Gambetta, Physical Review Letters **119**, 180509 (2017).
[53] A. Lowe, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, Physical Review Research **3**, 033098 (2021).
[54] P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, Quantum **5**, 592 (2021).
[55] A. Sopena, M. H. Gordon, G. Sierra, and E. López, Quantum Science and Technology **6**, 045003 (2021).
[56] E. Van Den Berg, Z. K. Minev, and K. Temme, Physical Review A **105**, 032620 (2022).
[57] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, and T. E. O'Brien, arXiv (2022), 10.48550/arXiv.2210.00921.
[58] S. Aaronson and D. Gottesman, Physical Review A **70**, 052328 (2004).
[59] H. Pashayan, O. Reardon-Smith, K. Korzekwa, and S. D. Bartlett, PRX Quantum **3**, 020361 (2022).
[60] B. Nachman, M. Urbanek, W. A. De Jong, and C. W. Bauer, npj Quantum Information **6**, 84 (2020).
[61] Matteo Robbiati, Alejandro Sopena, BrunoLiegiBastonLiegi, and Stefano Carrazza, "qiboteam/rtqem: Version 0.0.1," (2023).
[62] 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 European Physical Journal C **82** (2022), 10.1140/epjc/s10052-022-10328-7.
[63] "QuantWare," (2023).
[64] "Qblox Cluster," (2023).
[65] "qibolab_platforms_qrc," (2023).
[66] "IQM Quantum Computers," (2023).
[67] "Zurich Instruments Quantum Computing Control System," (2023).

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.

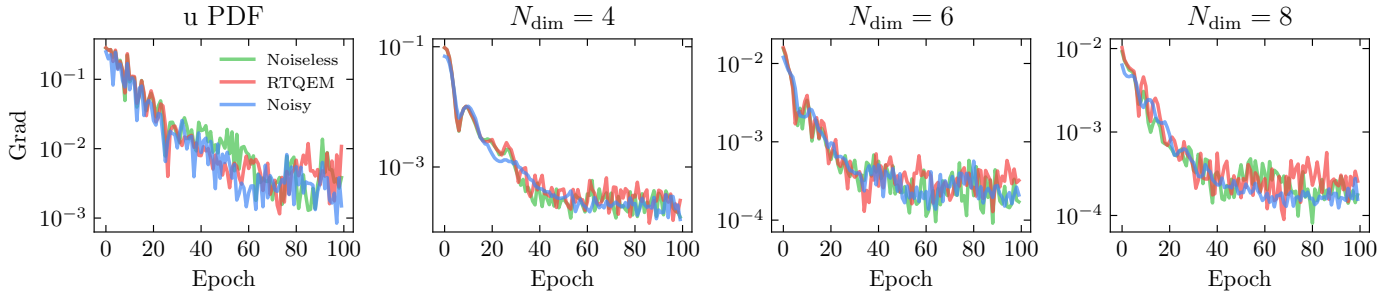

FIG. 9. Average gradients as function of the optimization epochs. A noiseless simulation (green lines) is compared with unmitigated noisy simulation (blue lines) and RTQEM (red lines) for $N_{\text{dim}} = 4, 6, 8$ from the left to the right plots.

Appendix B: Native gates

The native gates of the QRC superconducting quantum processors are $RX(\pm\pi/2)$, $RZ(\theta)$, and CZ gates [65]. They constitute a universal quantum gate set. These gates are compiled into microwave pulses following a specific set of rules [27]. 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,

$$U(\theta, \phi, \lambda) = RZ(\phi)RX(-\pi/2)RZ(\theta)RX(\pi/2)RZ(\lambda) \ . \tag{A1}$$