

Determining probability density functions with adiabatic quantum computing

Matteo Robbiati,^{1,2} Juan M. Cruz-Martinez,³ and Stefano Carrazza^{2,4,3,5}

¹*European Organization for Nuclear Research (CERN), Geneva 1211, Switzerland*

²*TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano, Milan, Italy*

³*CERN, Theoretical Physics Department, CH-1211 Geneva 23, Switzerland*

⁴*INFN, Sezione di Milano, I-20133 Milan, Italy*

⁵*Quantum Research Center, Technology Innovation Institute, Abu Dhabi, UAE*

The two main approaches to quantum computing are gate-based computation and analog computation, which are polynomially equivalent in terms of complexity, and they are often seen as alternatives to each other. In this work, we present a method for fitting one-dimensional probability distributions as a practical example of how analog and gate-based computation can be used together to perform different tasks within a single algorithm. In particular, we propose a strategy for encoding data within an adiabatic evolution model, which accommodates the fitting of strictly monotonic functions, as it is the cumulative distribution function of a dataset. Subsequently, we use a Trotter-bounded procedure to translate the adiabatic evolution into a quantum circuit in which the evolution time t is identified with the parameters of the circuit. This facilitates computing the probability density as derivative of the cumulative function using parameter shift rules.

Keywords: Analog Computing, Quantum machine learning; Hybrid computation; Variational quantum circuits; Optimization

I. INTRODUCTION

In the context of quantum computing, we are witnessing the development of various technologies, which can be categorized into two different but computationally equivalent approaches: Gate-Based Computation (GBC) and analog computation [1] (AQC).

These two approaches are often used to address very different types of problems. On one hand, many of the most well-known quantum computing query algorithms such as Shor's [2], Grover's [3] or Deutsch-Josza's [4] are formalized through the gate computation paradigm. Also in the field of Quantum Machine Learning (QML) [5, 6], the most common approach involves defining Parametric Quantum Circuits [7–9] that serve as variational models which are trained to perform the target tasks. On the other hand, AQC has been shown to be an effective tool for tackling optimization problems [10–14], in particular Quadratic Unconstrained Binary Optimization (QUBO) problems, which can be easily encoded within a system of interacting nearest-neighbours particles and represented in terms of Ising Hamiltonians.

In this work, we present an application where AQC and GBC can be used together in the context of QML, addressing different tasks within the process by exploiting their respective strengths. To showcase our proposed algorithm we tackle the determination of the underlying Probability Density Function (PDF) of a given one-dimensional dataset.

This is a problem that presents some very specific challenges. First, given a random sample of a distribution, a way of reconstructing the underlying distribution is to compute its Cumulative Distribution Function (CDF). An

accurate representation of a CDF requires a model which behaves monotonically with a target parameter, for which we exploit AQC. Then, given its CDF, the PDF can be determined by the derivative of the CDF. We face this second challenge through GBC (Fig. 1).

Let us then begin by considering the cumulative distribution of a sample. We first define a regression model based on adiabatic evolution [15] which encodes a generic one-dimensional function defined in a predefined bounded range as the time evolution of the expectation value of an arbitrary observable over the evolved state. This approach is sufficiently flexible to fit a large variety of functional forms and it can be easily set up so that boundary conditions and the monotonicity of the problem are automatically satisfied with a suitable definition of the adiabatic evolution. This final remark is particularly important when dealing with Cumulative Distribution Functions (CDF), which have to be monotonically increasing and defined between 0 and 1.

After achieving an acceptable CDF fit, the method projects the obtained adiabatic Hamiltonian into a quantum circuit representation using a Trotter-like decomposition [16]. This step opens the possibility to train and perform inference of the regression model on circuit-based quantum devices and therefore give us the possibility to extract the PDF of the sample as the derivative of the circuit using robust and well known Parameter Shift Rules (PSR) [17, 18].

The paper is organized as follows. In Sec. II we present the technical details of the probability density function estimation using the synergistic action of analog and gate-based quantum computing. The Sec. III presents validation results for multiple examples and a comparison with classical Kernel Density Estimation (KDE) methods to ver-

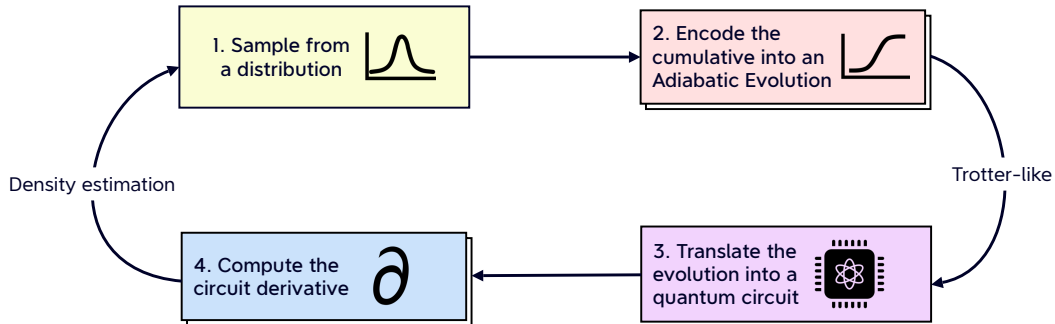


FIG. 1. Schematic representation of the proposed Quantum Adiabatic Machine Learning (QAML) algorithm. A cumulative distribution function is encoded into an adiabatic evolution, which is then translated into a quantum circuit and differentiated with respect to the target values in order to approximate the probability density function.

ify the robustness of our algorithm in terms of accuracy. Finally, in Sec. V we draw our conclusion and outlook.

II. METHODOLOGY

In this section we describe the procedure implemented for the determination of probability density functions. The algorithm is separated in two steps: the approximation of an empirical cumulative distribution function using adiabatic quantum evolution in a discrete time-grid as a regression model, and subsequently, the determination of the probability density function through the trotter-like quantum circuit representation obtained from the adiabatic Hamiltonian. In the same section we generalize the evolution time to a continuous variable.

In Sec. II A and Sec. II B 2 we introduce the choice of using an adiabatic evolution to fit a cumulative distribution. And in Sec. II C 2 we translate the obtained operator into a quantum circuit, facilitating the computation of its derivatives.

A. Model regression with adiabatic quantum evolution

Given a function $F(t)$, one-dimensional in input and output, we build a regression model by selecting an observable \mathcal{O} such that there are two Hamiltonians, H_0 and H_1 , for which the expectation value of \mathcal{O} over the ground states of H_0 and H_1 correspond to the two points between which we want to learn the function F .

Therefore, we interpret the regression problem as the procedure of building a time dependent Hamiltonian $H(t)$, such that its ground state $|\psi(t)\rangle$ at time t satisfies

$$\langle\psi(t)|\mathcal{O}|\psi(t)\rangle = F(t). \quad (1)$$

From now on, for simplicity, we shorten the l.h.s. of the expression (1) with $\langle\mathcal{O}\rangle_t$. We construct this Hamiltonian implementing an adiabatic evolution

$$H(t) = [1 - s(t; \boldsymbol{\theta})]H_0 + s(t; \boldsymbol{\theta})H_1, \quad (2)$$

governed by the parametric scheduling function $s(t; \boldsymbol{\theta})$, where t has to be defined in $[0, 1]$. The problem is then reduced to finding the right set of parameters $\boldsymbol{\theta}$ such that $\langle\mathcal{O}\rangle_t$ during the adiabatic evolution of the state $|\psi(t)\rangle$ approximates the target function.

Note that the choice of $s(t, \boldsymbol{\theta})$ is fundamental to guarantee the monotonicity of the target function.

B. Learning empirical cumulative density functions

1. Adiabatic evolution setup

The presented framework can be applied to the problem of fitting a cumulative distribution function $F(t)$, with $t \in [0, T]$. This can be done if two requirements are satisfied: the model has to be strictly monotonically increasing in t and the extreme values of the function are set to be $F(0) = 0$ and $F(T) = 1$.

The second condition can be fulfilled by appropriately selecting the Hamiltonians H_0 and H_1 , as well as the observable \mathcal{O} . In particular, since we focus on one-dimensional distributions, and we treat here the introductory case of one qubit, a proper choice can be $\mathcal{O} = \sigma_z$, $H_0 = \sigma_x$ and $H_1 = -\sigma_z$, where σ_x and σ_z correspond to the Pauli X and Pauli Z matrices respectively. This choice satisfies the boundary conditions of the problem $\langle\mathcal{O}\rangle_0 = 0$ and $\langle\mathcal{O}\rangle_T = 1$. Secondly, the monotonicity of the function can be ensured by implementing a scheduling function which is monotonic itself. This final remark, together with the appropriate Hamiltonians definition, make adiabatic evolution

an extremely effective model for approximating a CDF. In this work, we use as scheduling function a polynomial of degree p :

$$s(t; \theta) = \frac{1}{\eta} \sum_{i=1}^p \theta_i t^i, \quad \text{with} \quad \eta = \sum_{i=1}^p \theta_i, \quad (3)$$

whose monotonicity is guaranteed by requiring positive coefficients θ_i for $t \in [0, 1]$. If one prefers to define a more flexible scheduling function, such as a neural network, another viable option is to penalize the model during training when negative fluctuations of the predictions are recorded.

2. Training of the adiabatic evolution

In order to train a parametric adiabatic evolution to approximate a function we make use of Qibo [19–22], which is an open-source full-stack framework for quantum computing. Qibo provides the possibility to execute analog quantum computing models that can be symbolically defined through an interface based on SymPy [23]. These models can then be translated into circuits and executed through trotterization. In particular, a second-order Time-Evolving Block Decimation (TEBD) method is implemented (see Eq. (62) of Ref. [16]). This approximation, once chosen a time step $d\tau$ presents an error of the order $O(d\tau^3)$ per time step [16]. Considering a total evolution time T divided into $N = T/d\tau$ steps, the total error of the approximation becomes $O(d\tau^2)$. Further details can be found in the official Qibo documentation [24].

The use of Qibo allows us to execute our algorithm on self-hosted superconducting quantum devices (see Sec. IV), however by this choice we are limited to a gate-based paradigm, which forces us to consider a discretized case of the time evolution. We address this limitation in Sec. II C 1, by implementing an approximation that allows us to extend from the discrete case to the continuous. This approximation would not be necessary if using analog devices or simulators. Among the simulation tools we mention QuTip [25] and HOQST [26] for handling open quantum systems, and QuantumAnnealing.jl [27] for quantum annealing algorithms.

The procedure follows a supervised machine learning strategy: at first, we generate a sample of random variables $\{x\}$ following a chosen distribution ρ , and we calculate the empirical CDF of the sample $\{F\}$. This step is equivalent to constructing a cumulative histogram of the data, where we select N_{train} bins identified with the evolution times controlling the scheduling function. Each pair of (x_j, F_j) values are mapped into $(\tau_j, \langle \mathcal{O} \rangle_{\tau_j})$, where $\tau_j = t_j/T$ is the evolution time corresponding to the j -th step of the discretized adiabatic evolution normalized with respect to the final time T and $\langle \mathcal{O} \rangle_{\tau_j}$ is the expectation

value of the target observable over the ground state of $H(\tau_j)$ as introduced in Eq. (1).

In this last step, we apply a transformation to the domain of the variable x , so that both x and τ are now defined in the interval $[0, 1]$. This allows us to define a standardized framework for each analyzed variable. The original distribution can be easily reconstructed by applying the inverse transformation.

Once the training set is defined and the scheduling function is initialised using an initial set of parameters θ_0 , we execute the adiabatic evolution of the state $|\psi(\tau)\rangle$ from $|\psi(0)\rangle$ to $|\psi(1)\rangle$ and we collect all the $\langle \mathcal{O} \rangle_{\tau_j}$ values during the evolution.

We define a mean-squared error loss function J for estimating the quality of the fit:

$$J = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} [F_j - \langle \mathcal{O} \rangle_{\tau_j}]^2, \quad (4)$$

where $\langle \mathcal{O} \rangle_{\tau_j}$ depends on the variational parameters because the evolved state on which it is calculated follows an evolution governed by the parametric scheduling $s(\tau, \theta)$.

We finally perform the training of the model by optimizing the parameters θ using a selected optimizer. In this work we make use of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [28], which is one of the optimizers provided by Qibo. We remark that any optimizer can be used since the approach is totally agnostic under this aspect.

In Fig. 2 we show and describe an example of the algorithm presented above, to which we refer from now on as Quantum Adiabatic Machine Learning (QAML).

C. Deriving probability functions from quantum circuits

In the following we generalize the presented model to accept any continuously sampled time $t \in [0, T]$ and we use a quantum circuit representation to compute the derivative of the approximated CDF with respect to the target variable.

1. Generalizing the evolution to any time

The procedure presented in the previous paragraphs can be interpreted as the evolution product of a series $\{H(\tau_j)\}$ of Hamiltonians corresponding to the adiabatic Hamiltonian (2) at fixed evolution time steps $\{\tau_j = t_j/T\}$, discretized according to $d\tau = (t_j - t_{j-1})/T$, the time step of the adiabatic evolution. Each of these Hamiltonians can be associated to a *local* time evolution operator $U(\tau_j)$ which evolves $|\psi(\tau_{j-1})\rangle$ to $|\psi(\tau_j)\rangle$. More generally, we can obtain

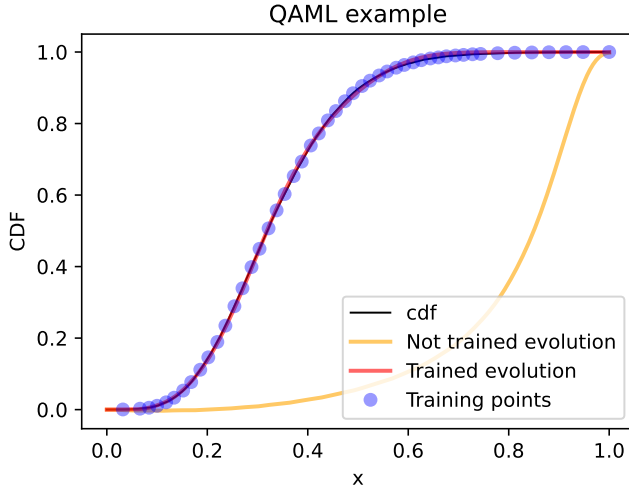


FIG. 2. Example of Quantum Adiabatic Machine Learning (QAML). We select N_{train} data from a sample of points picked up from a Gamma distribution. The training labels (blue points) lay on the empirical CDF of the sample (black line). The untrained sequence of $\langle \mathcal{O} \rangle$ (yellow line) is compared with the values after the QAML training (red line). The scheduling function is the one presented in Eq. (3) with $\rho = 12$.

any state $|\psi(\tau_n)\rangle$ by sequentially applying n operators:

$$|\psi(\tau_n)\rangle = \prod_{j=1}^n \mathcal{T}U(\tau_j) |\psi(\tau_0)\rangle := \mathcal{C}(\tau_n) |\psi(\tau_0)\rangle, \quad (5)$$

where we sum up the sequential action of the $U(\tau_j)$'s into a single unitary operator $\mathcal{C}(\tau_n)$, which evolves the initial state $|\psi(\tau_0)\rangle$ to the one at time $\tau_n = n d\tau$ and depends on the time-ordering operator \mathcal{T} . From now on, we refer to $H(\tau_j)$, $U(\tau_j)$ and $\mathcal{C}(\tau_j)$ as H_j , U_j and C_j for simplicity.

In order to compute the state at any value of τ outside of the discrete simulated time steps of the adiabatic evolution $\{\tau_j\}$ it is necessary to take the continuous limit. We start by considering one of the intermediate elements of the product introduced in Eq. (5)

$$U_j = e^{-id\tau H_j}, \quad \text{with} \quad H_j = \begin{pmatrix} s_j & 1 - s_j \\ 1 - s_j & -s_j \end{pmatrix}, \quad (6)$$

where s_j is the value of the scheduling at evolution time $\tau_j = j d\tau$. This instantaneous form of the adiabatic Hamiltonian operator can be diagonalized as D_j using a matrix P_j such that $H_j = P_j D_j P_j^{-1}$, where:

$$P_j = \Lambda_j \begin{pmatrix} 1 & \frac{s_j - \lambda_j}{1 - s_j} \\ \frac{\lambda_j - s_j}{1 - s_j} & 1 \end{pmatrix}, \quad D_j = \begin{pmatrix} \lambda_j & 0 \\ 0 & -\lambda_j \end{pmatrix}, \quad (7)$$

and Λ_j is the appropriate (τ dependent) normalization constant. The absolute value of the eigenvalues of the Hamiltonian is $\lambda_j = \sqrt{2s_j^2 - 2s_j + 1}$.

We now use this decomposition to write \mathcal{C}_n in terms of the diagonal form of the Hamiltonian:

$$\mathcal{C}_n = \prod_{j=0}^n P_j e^{-iD_j d\tau} P_j^{-1}. \quad (8)$$

If we now take the limit $d\tau \rightarrow 0$, we have $H_j \rightarrow H_{j-1}$ and thus $P_j \rightarrow P_{j-1}$. We have verified that for smoothly behaved scheduling function adjacent elements in the sequence tend to the identity $P_j^{-1} P_{j-1} \rightarrow I$ with an error proportional to the time step $d\tau$. The Eq. (8) simplifies to

$$\mathcal{C}_n = P_n \exp \left\{ -i \sum_{j=0}^n D_j d\tau \right\} P_0^{-1}. \quad (9)$$

Furthermore, in the limit of $d\tau \rightarrow 0$, $n \rightarrow \infty$ and the sum in the above equation becomes an integration in $d\tau$ between the initial and final steps in the evolution.

$$\mathcal{C}_t = P_t \exp \left\{ -i \int_0^{t/T} D_j d\tau \right\} P_0^{-1}, \quad (10)$$

where we now indicate with P_t and P_0 the diagonalization matrices corresponding respectively to the last and the first evolution operators we must apply to H_0 's ground state in order to obtain the evolved state at an arbitrary time t , dropping the discretization requirement.

The proposed approximation only holds when considering the adiabatic regime, implemented through a small time step and a slowly varying scheduling function. More in general, one can construct a more complete and arbitrary accurate representation of U_t exploiting the Magnus Expansion (ME) [29, 30]. Combining a proper choice of the ME order with an arbitrary small trotterization error (defined by the Trotter-Suzuki order and step size) one can construct an arbitrarily accurate approximation of U_t as discussed in [31].

2. Circuit representation

Let us now implement the unitary operator \mathcal{C}_t as a quantum circuit, using a gate decomposition which is useful for calculating the derivatives of the circuit with respect to its variational parameters.

To that end we write \mathcal{C}_t in terms of rotational gates, with the aim of using well-known parameter shift rules [17, 18, 32] on them. The choice of this differentiation method relies on its robustness to noise and acknowledging that in the one-dimensional case, the decomposition we propose is simple and computationally lightweight.

Since any unitary operator $\mathcal{U} \in SU(2)$ can be written as a combination of three rotations [33] we choose:

$$\mathcal{U} \equiv R_z(\phi) R_x(\theta) R_z(\psi), \quad (11)$$

where the three angles (ϕ, θ, ψ) can be computed as function of the matrix element of the operator \mathcal{C} :

$$\begin{cases} \phi = \pi/2 - \arg(c_{01}) - \arg(c_{00}), \\ \theta = -2 \arccos(|c_{00}|), \\ \psi = \arg(c_{01}) - \pi/2 - \arg(c_{00}). \end{cases} \quad (12)$$

In (12) the matrix elements c_{00} and c_{01} depend on the values of the scheduling s and on the eigenvalues λ of H_t . This dependence can be written more explicitly:

$$c_{0j} = \frac{1-s}{s\sqrt{\lambda(\lambda-s)}} \left\{ \cos \mathcal{I} \left(1 + (-1)^j \frac{\lambda-s}{1-s} \right) + i \sin \mathcal{I} \left(1 - (-1)^j \frac{\lambda-s}{1-s} \right) \right\}, \quad (13)$$

with $\mathcal{I} = \int_0^\tau \lambda(t') dt'$, $s = s(\tau)$, $\lambda = \lambda(\tau)$ and $\tau = t/T$.

Note that the construction of the circuit is completely independent of the choice of scheduling function. Depending on this choice, \mathcal{I} may be computed analytically.

With this, we define a method to build a quantum circuit which, for fixed evolution parameters θ and time t , returns the ground state of H_t , which can be used to compute the requested $\langle \mathcal{O} \rangle_t$.

3. From the CDF to the PDF

The circuit representation of the operator \mathcal{C}_t allows us to reconstruct our original target function $F(t)$ introduced in Sec. II A by applying the circuit to a state prepared as $|\psi(0)\rangle$ and then measuring the chosen observable (σ_z in this work) over the obtained final state.

As previously said, our example case has been that in which the target function $F(t)$ correspond to the empirical CDF of some arbitrary distribution ρ . By imposing monotonicity and pinning the initial and final points we ensure that its first derivative corresponds to the PDF of the same distribution.

For a one-dimensional distribution we have then:

$$\frac{dF(t)}{dt} = \frac{d}{dt} \langle \psi(0) | \mathcal{C}_t^\dagger \sigma_z \mathcal{C}_t | \psi(0) \rangle. \quad (14)$$

In the context of quantum computing, as previously anticipated, we can take advantage of what is usually known as Parameter Shift Rule (PSR) [17, 18] which allows us to take the derivative of the expectation of an observable such as Eq. (14) by simply evaluating the circuit after shifting the parameters with respect to which we are taking the derivative. We are using specifically the formula presented in [17], for circuits based on rotations. Note that in this case we have limited ourselves to gates in which the parameter appears only once, but more complicated forms can also be utilized [34].

With this we arrive to the final formula of the PDF in terms of the original circuit for an arbitrary value of t :

$$\rho(t) = \text{PSR} \left[\langle \psi(0) | \mathcal{C}_t^\dagger \sigma_z \mathcal{C}_t | \psi(0) \rangle \right]. \quad (15)$$

III. VALIDATION

In the following we test the presented algorithm by drawing samples from a known distribution, building the circuit and reconstructing the original probability function. All results for this section are summarized in Table I.

A. Sampling known distributions

In order to validate and test the procedure, we select two known distributions: a Gamma distribution and a Gaussian mixture of two Gaussian distributions. For each case, we generate a representative sample of dimension N_{sample} and fit the resulting empirical CDF using the approach described in Sec. II A. We then derive the PDF with the procedure introduced in Sec. II C 3 and compare the results with the original distribution. We repeat this exercise for every example by using quantum simulation on classical hardware with exact state-vector representation and with shot-noise.

In these examples the adiabatic evolution training is set to stop once a given threshold value J_{thresh} of the loss function (4) is reached. Furthermore, every sample is rescaled to be between 0 and 1 according to the definition of τ introduced in Sec. II B 2. In all cases the adiabatic evolution is run from $\tau = 0$ to $\tau = 1$ with $d\tau = 0.002$. We define the scheduling function as a polynomial of order p following the ansatz in Eq. (3).

We start by drawing samples from a Gamma distribution, defined as

$$\rho(x; \alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, \quad (16)$$

with $\alpha = 10$ and $\beta = 0.5$. We take $N_{\text{samples}} = 5 \cdot 10^4$ points and train the scheduling function until a target precision of $J_{\text{thresh}} = 10^{-5}$ is reached.

We repeat the procedure both with and without shot-noise. In the case of simulations with shot-noise, we construct the final value of our predictions by repeating the calculation of the expectation value of σ_z twenty times. The collected results are then used to define the prediction and its uncertainty as the mean and standard deviation of the obtained values, respectively. The discussed results are shown in Tab. I.

The results of the training can be seen in the first row and left column of Fig. 3, where we plot the empirical CDF (black line) together with both the exact (red line)

Fit function	N_{sample}	p	J_f	$\text{MSE}_{\text{CDF}}^{\text{exact}}$	$\text{MSE}_{\text{CDF}}^{\text{shots}}$	$\text{MSE}_{\text{PDF}}^{\text{exact}}$	$\text{MSE}_{\text{PDF}}^{\text{shots}}$	$\text{KL}_{\text{PDF}}^{\text{shots}}$
Gamma	$5 \cdot 10^4$	25	$2.9 \cdot 10^{-6}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$9 \cdot 10^{-4}$	$2 \cdot 10^{-3}$	$4 \cdot 10^{-3}$
Gaussian mix	$2 \cdot 10^5$	30	$4.4 \cdot 10^{-6}$	$9 \cdot 10^{-6}$	$9 \cdot 10^{-6}$	$2 \cdot 10^{-3}$	$4 \cdot 10^{-3}$	$6 \cdot 10^{-3}$
t	$5 \cdot 10^4$	20	$2.1 \cdot 10^{-6}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	$5 \cdot 10^{-3}$
s	$5 \cdot 10^4$	20	$7.9 \cdot 10^{-6}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$1 \cdot 10^{-2}$	$1 \cdot 10^{-2}$	$4 \cdot 10^{-3}$
y	$5 \cdot 10^4$	8	$3.7 \cdot 10^{-6}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$9 \cdot 10^{-4}$	$2 \cdot 10^{-3}$	$2 \cdot 10^{-3}$

TABLE I. Summary of the results. From left to right, we show the number of points of the datasets, the number of variational parameters of the adiabatic evolution models, the best loss function value registered during the training, the MSE metric values respectively comparing the CDF estimation in exact simulation, the CDF estimation in shot-noise simulation, the PDF estimation in exact simulation and the PDF estimation in shot-noise simulation. When considering shot-noise simulation, each estimation contributing to the MSE has been calculated as mean value of 20 estimates, each of these obtained collecting $N_{\text{shots}} = 2 \cdot 10^5$ shots of the quantum circuit. The MSE valued for the Gamma and the Gaussian mixture examples are calculated using five hundred points equispaced in $[0, 1]$, while the HEP results are calculated considering the values extracted from a histogram representation of the data setting $N_{\text{bins}} = 34$. In the last column, we show the value of the Kullback-Leibler divergence between the estimated distribution in case of $N_{\text{shots}} = 2 \cdot 10^5$ and the theoretical distribution.

and shot-noise (blue and yellow lines) simulation, and a histogram of the data. In the second row and left column of Fig. 3 we show the PDF obtained by taking the derivative of the circuit compared it with the original distribution, as well as the ratio with respect to the target labels. We find that while the exact simulation achieves an accuracy of a few percent everywhere but the tails of the distribution, when shot-noise is enabled it is necessary to go beyond $N_{\text{shots}} = 10^5$ to achieve single-digit precision. This is illustrated by the two shadowed bands for $N_{\text{shots}} = 2 \cdot 10^5$ (blue), and a lower value of $N_{\text{shots}} = 2 \cdot 10^4$ (yellow) which has a worse than 10% precision everywhere.

In order to test the algorithm with a more complicated example we also sample from a Gaussian mixture

$$\rho(x; \vec{\mu}, \vec{\sigma}) = 0.6\mathcal{N}(x; \mu_1, \sigma_1) + 0.4\mathcal{N}(x; \mu_2, \sigma_2), \quad (17)$$

with $\vec{\mu} = (-10, 5)$ and $\vec{\sigma} = (5, 5)$. From this distribution we take $N_{\text{sample}} = 5 \cdot 10^5$ points to generate the training sample. The results corresponding to this second target are shown in the right column of Fig. 3 and follow the same graphical conventions presented above.

To quantify the accuracy of our predictions we define a Mean Squared Error (MSE) metric

$$\text{MSE} = \frac{1}{N} \sum_{j=0}^N (y_{j,\text{meas}} - y_{j,\text{pred}})^2, \quad (18)$$

where $y_{j,\text{meas}}$ and $y_{j,\text{pred}}$ correspond to the target label and the model prediction associated to a specific variable x_j . The Eq. (18) is written as function of a general variable y , which is then deployed as the CDF and the PDF predictions in what follows. In addition, we also compute the Kullback-Leibler divergence

$$\text{KL} = \sum_{i=1}^N \left[y_{i,\text{pred}} \log \left(\frac{y_{i,\text{pred}}}{y_{i,\text{meas}}} \right) \right] \quad (19)$$

for the PDFs computed in the case of shot-noise simulation with $N_{\text{shots}} = 2 \cdot 10^5$ shots. We collect our results in Tab. I.

B. Density estimation of simulated high energy physics data

In the previous case we were training the circuit using a sample from a known distribution, we now address the more complex case of learning an unknown distribution. We consider the particle physics process involving top and anti-top quark pair production ($pp \rightarrow t\bar{t}$), for which we choose the cross section differential on the rapidity y and the logarithms of the Mandelstam variables $-\log(-t)$ and $-\log s$. This choice is motivated by the following reasons: on one hand, they present a real-world scenario in high energy physics (HEP), stress-testing the method, and on the other hand, it provides a potential use-case for the methods presented in this paper.

While in general one would train directly on the output of a Monte Carlo event generator, in our test case the data sampling is obtained from a separated hybrid classical-quantum model called Style-qGAN [35]. The Style-qGAN has been trained with 10^5 events for $pp \rightarrow t\bar{t}$ production at a center of mass of $\sqrt{s} = 13$ TeV computed at Leading Order.

In Fig. 4 we show the results for the training of the circuit (the CDF on the top row) and its derivative (the PDF on the bottom row), following the same graphical conventions introduced in the previous section.

In Table I we summarize the results for all examples tested in this section. For each model we describe the final configuration and fit accuracy by calculating the MSE values for both the CDF and the PDF estimations.

We find the achieved level of quality is satisfactory for all tested distributions. We also observe that $N_{\text{shots}} = 2 \cdot 10^5$ shots provide sufficient statistics to achieve a precision in

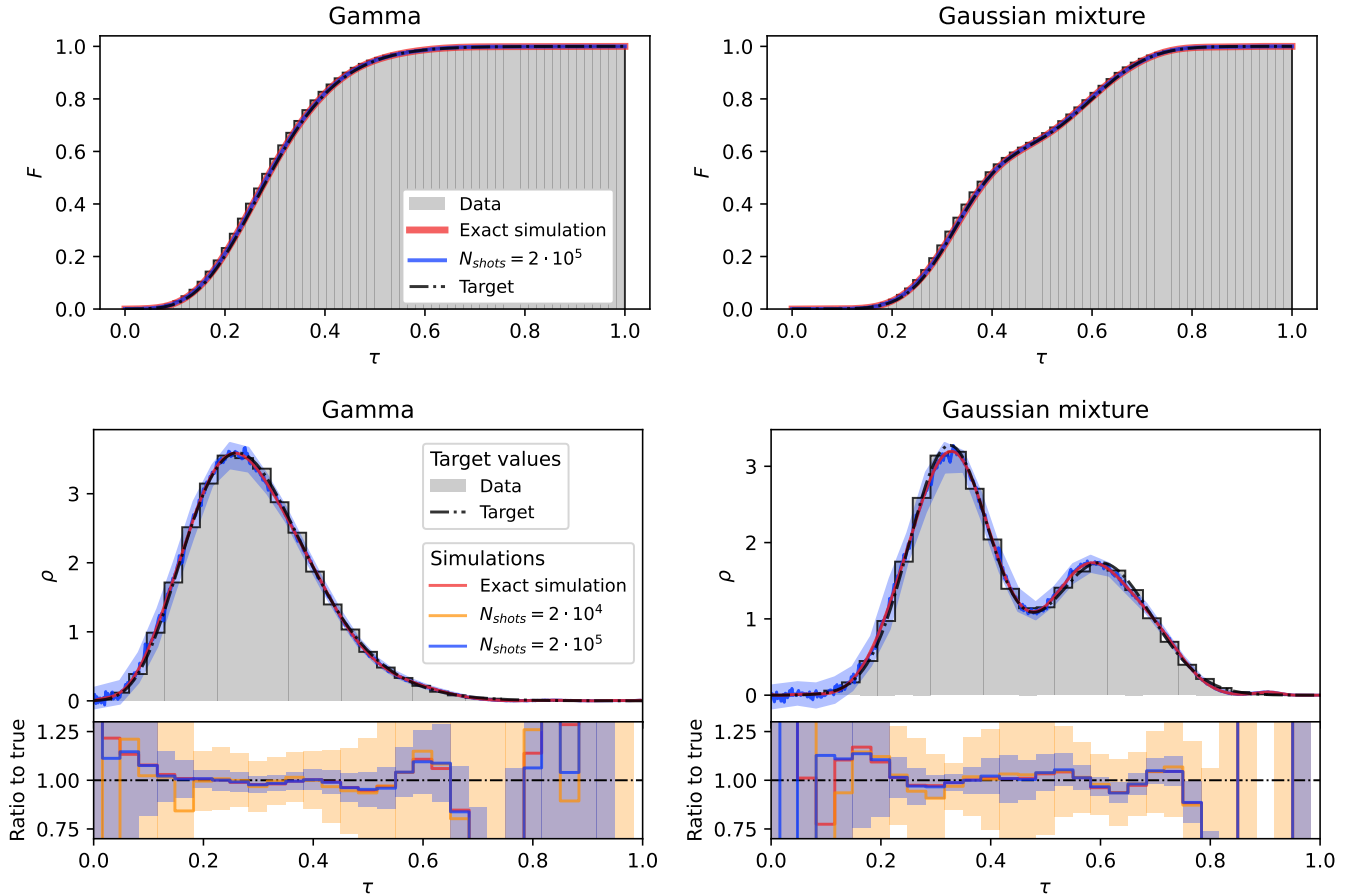


FIG. 3. Top row: CDF fit via QAML procedure. The target CDF (dashed black line) is compared with QAML predictions obtained via exact simulation (red line) and simulation with shot-noise (blue line). An histogram of the data is shown in grey. Bottom row: comparison between the target PDF and the result of the derivative of the trained circuit. Once again the target labels (dashed black line) are compared with the data histogram (grey), with QAML exact simulation results (red line) and QAML shot-noise simulations with $N_{shots} = 2 \cdot 10^4$ and $N_{shots} = 2 \cdot 10^5$ (yellow and blue lines respectively). These same results are also shown in the form of a ratio between the target labels and the predictions via QAML in the lowest part of the figures. While representing the PDFs, only one simulation with shot-noise is drawn ($N_{shots} = 2 \cdot 10^5$). All the shot-noise simulations curves are represented together with a 1 σ confidence belt calculated repeating $N_{runs} = 20$ the predictions with fixed trained model.

the range of 4-10% (see ratio plots in the bottom part of Fig. 3 and Fig. 4). In the case of the HEP based data, this level of precision is only achieved near the distribution peaks. The precision (and the accuracy) deteriorates as we move towards the tails where the differential cross section approaches zero, a behavior consistent with classical approaches a common challenge when fitting distributions [36].

C. Benchmark with KDE methods

In this section we benchmark our results with a state-of-art Kernel Density Estimation method provided by `scikit-learn` [37]. While we are optimistic about the future of quantum technologies and the potential utility

of the proposed method, we stress that this analysis is intended here to only verify the robustness of the results in terms of accuracy, and not to propose this method as an alternative to classical techniques. This kind of stress-testing is necessary in order to have confirmation of the validity of the proposed method once faster and more accurate hardware is obtained.

We compare our exact-simulation results with the predictions computed using the `sklearn.neighbors.KernelDensity` method selecting *top-hat* and *exponential* kernels. The first has been chosen for its computational efficiency, while the second is more suitable when dealing with tailed distributions. The kernel bandwidth has been hyper-optimized using a Tree of Parzen Estimators (TPE) on a random grid of one thousand points sampled from the interval

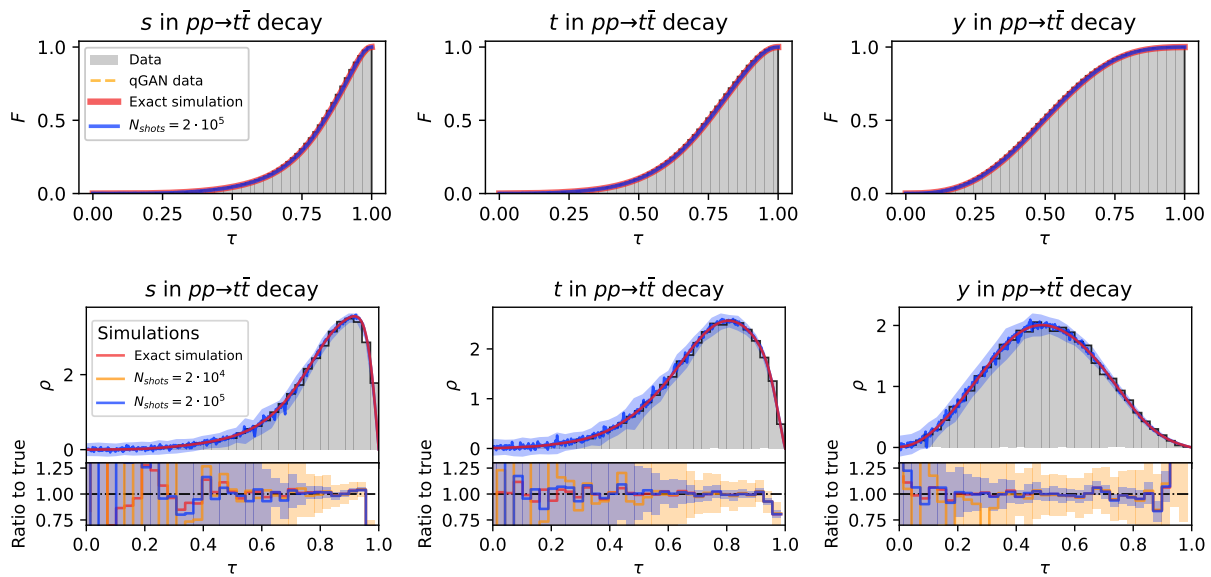


FIG. 4. Top row: CDF fit via QAML procedure considering the HEP targets. The target CDF (dashed black line) is compared with QAML predictions obtained via exact simulation (red line) and simulation with shot-noise (blue line). An histogram of the data is shown in grey. Bottom row: comparison between the target PDF and the result of the derivative of the trained circuit. Once again the target labels (dashed black line) are compared with the data histogram (grey), with QAML exact simulation results (red line) and QAML shot-noise simulations with $N_{\text{shots}} = 2 \cdot 10^4$ and $N_{\text{shots}} = 2 \cdot 10^5$ (yellow and blue lines respectively). These same results are also shown in the form of a ratio between the target labels and the predictions via QAML in the lowest part of the figures. While representing the PDFs, only one simulation with shot-noise is drawn ($N_{\text{shots}} = 2 \cdot 10^5$). All the shot-noise simulations curves are represented together with a 1σ confidence belt calculated repeating $N_{\text{runs}} = 20$ the predictions with fixed trained model.

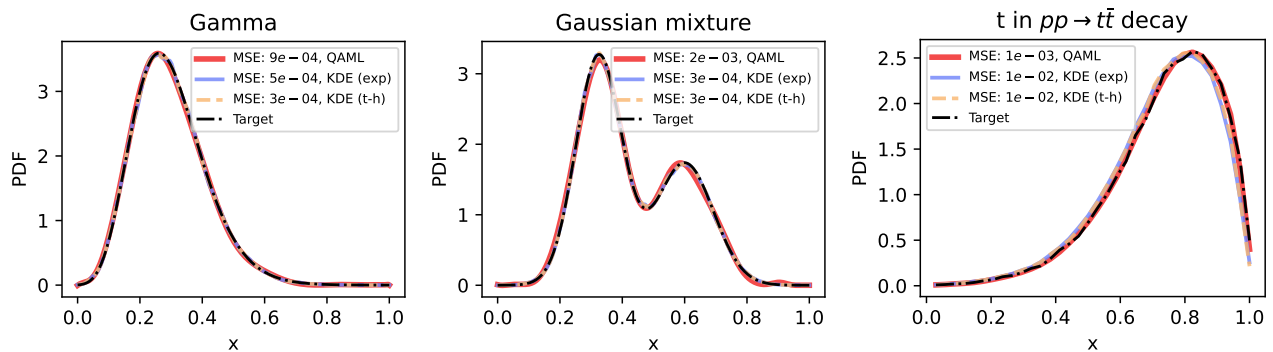


FIG. 5. Comparison between our QAML trained results (red lines) and the proposed solutions of 1D Kernel Density Estimation method provided by `scikit-learn` using *top-hat* (orange lines) and *exponential* (blue lines) kernels. All the approximations are compared with the target values (black lines).

$[10^{-5}, 1]$. The hyper-optimization has been performed using `hyperopt` [38].

We compute this benchmark in order to check whether our algorithm is competitive in terms of accuracy with respect to the most commonly used classical tools. We test both the QAML procedure and the KDE algorithm on the Gamma distribution, the Gaussian mixture and the t Mandelstam variable, which is the one corresponding to our best result in the HEP examples. We show these results in Fig. 5, where our predictions are promisingly close to

those obtained through KDE estimation. Additionally, in the case of the variable t , which is defined on a more critical domain due to the chosen binning, QAML seems to offer a more accurate fit to the density profile.

Considering the execution time of the two algorithms, they present very different characteristics: on one hand, QAML suffers the optimization time, which is necessary to perform the density estimation. On the other hand, when the training is concluded, computing the PDF is extremely lightweight. Instead, in the case of the KDE, the opposite

considerations can be made: the hyper-optimization is relatively light, but then the PDF evaluation depends on the bandwidth size.

IV. HARDWARE

In order to assess the performance of the model in real quantum hardware we use a 5-qubits superconducting chip hosted in the Quantum Research Centre (QRC) of the Technology Innovation Institute (TII). We calculate the predictions for the values of the CDF using the best parameters obtained through the training with the shot-noise simulation whose results are presented in Tab. I. We take into account the Gamma distribution defined in Eq. (16) and we do not apply error mitigation techniques to the hardware, in order to explore the potentialities of the bare chips.

We consider $N_{\text{data}} = 25$ points equally distributed in the target range $[0, 1]$ and for each of these we perform $N_{\text{runs}} = 10$ predictions executing $N_{\text{shots}} = 1000$ times the circuit on the quantum hardware. Using these data, we calculate the final predictors and their uncertainties as mean and standard deviation over the N_{runs} results. We also calculate the MSE introduced in Eq. (18) to evaluate the fit accuracy.

In order to study how the CDF predictions deteriorate when executed in hardware and to study the dependence of this deterioration on how the qubits are tuned, we repeat this procedure for each of the five qubit of the device. The results are shown in Fig. 6 and are in agreement with Tab. II, where we report the assignment fidelities [39] of the qubits and the calculated MSE values.

Qubit ID	Assignment Fidelity	MSE
0	0.926	$1.3 \cdot 10^{-2}$
1	0.886	$1.4 \cdot 10^{-2}$
2	0.953	$3.4 \cdot 10^{-3}$
3	0.952	$2.7 \cdot 10^{-3}$
4	0.707	$1.3 \cdot 10^{-1}$

TABLE II. Prediction deployed on superconducting chip. For each qubit of the device we show the assignment fidelity at the moment of the execution and the MSE values.

The quantum hardware control is performed using QiboLab [21, 40] and the qubits are characterized and calibrated executing the Qibocal's routines [22, 41].

V. CONCLUSION

In this work we presented a proof-of-concept application of quantum machine learning which makes use of both

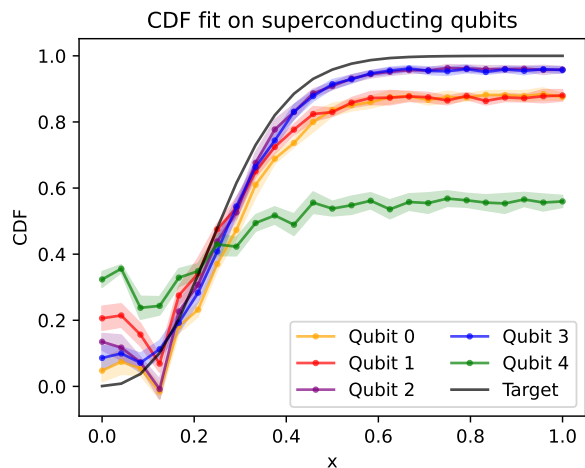


FIG. 6. $N_{\text{runs}} = 10$ predictions are performed for $N_{\text{data}} = 25$ points in the target range $[0, 1]$ using each qubit of a 5-qubits device hosted in the QRC.

analog and gate-based quantum computation addressing different tasks within the same problem. Namely, we introduced an algorithm for the determination of probability density functions. We first define a mechanism to use adiabatic evolution as a regression model for the fit of the cumulative density function of a sample. The adiabatic evolution model is then represented by the Trotterization of the adiabatic Hamiltonian in terms of a quantum circuit. The probability density profile is then calculated by applying parameter shift rules to the obtained circuit. This method allows the usage for training and inference of quantum devices designed for both annealing and circuit-based technologies. The numerical results obtained and presented in Sec. III show successful applications of the methodology for predefined PDFs and empirical distributions obtained from high-energy particle physics observables. In the same section we compare the presented methodology with classical Kernel Density Estimations techniques, demonstrating that the proposed method can yield satisfactory results when compared to state-of-art algorithms. Finally, in Sec. IV we deployed the trained models on to a superconducting bare device, showing interesting results even without applying any quantum error mitigation algorithm.

All numerical results have been obtained using Qibo [42], a full-stack and open source framework for quantum computing, and are publicly available in [43]. Further possible developments include the generalization of this method for the simultaneous determination of multi-dimensional probability density distributions, the deployment of the full training procedure on quantum devices and the possibility to use real quantum annealing for the optimization of the regressing model parameters.

ACKNOWLEDGMENTS

SC thanks the TH hospitality during the elaboration of this manuscript.

Funding: This project is supported by CERN's Quantum Technology Initiative (QTI). MR is supported by CERN doctoral program.

Author contributions: All authors contributed equally to the elaboration of this manuscript.

Data Availability: The code to reproduce the simulations can be found at [44].

DECLARATIONS

Conflict of interest: The authors declare no conflict of interest.

Open access: This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <https://creativecommons.org/licenses/by/4.0/>.

-
- [1] T. Albash and D. A. Lidar, Adiabatic quantum computation, *Reviews of Modern Physics* **90**, 10.1103/revmodphys.90.015002 (2018).
- [2] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM Journal on Computing* **26**, 1484–1509 (1997).
- [3] L. K. Grover, A fast quantum mechanical algorithm for database search (1996), arXiv:quant-ph/9605043 [quant-ph].
- [4] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* **439**, 553 (1992).
- [5] M. Schuld, I. Sinayskiy, and F. Petruccione, An introduction to quantum machine learning, *Contemporary Physics* **56**, 172–185 (2014).
- [6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature* **549**, 195 (2017).
- [7] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, *Quantum Science and Technology* **4**, 043001 (2019).
- [8] 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].
- [9] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, *Nature Reviews Physics* **3**, 625–644 (2021).
- [10] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, Quantum annealing for industry applications: introduction and review, *Reports on Progress in Physics* **85**, 104001 (2022).
- [11] J.-N. Zaech, A. Liniger, M. Danelljan, D. Dai, and L. V. Gool, Adiabatic quantum computing for multi object tracking (2022), arXiv:2202.08837 [cs.CV].
- [12] E. Pelofske, A. Bäertschi, and S. Eidenbenz, Quantum annealing vs. qaoa: 127 qubit higher-order ising problems on nisq computers, in *High Performance Computing* (Springer Nature Switzerland, 2023) p. 240–258.
- [13] P. Date and T. Potok, Adiabatic quantum linear regression, *Scientific Reports* **11**, 10.1038/s41598-021-01445-6 (2021).
- [14] B. Tasseff, T. Albash, Z. Morrell, M. Vuffray, A. Y. Lokhov, S. Misra, and C. Coffrin, On the emerging potential of quantum annealing hardware for combinatorial optimization (2022), arXiv:2210.04291 [math.OC].
- [15] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution (2000).
- [16] S. Paeckel, T. Köhler, A. Swoboda, S. R. Manmana, U. Schollwöck, and C. Hubig, Time-evolution methods for matrix-product states, *Annals of Physics* **411**, 167998 (2019).
- [17] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, *Physical Review A* **98**, 10.1103/physreva.98.032309 (2018).
- [18] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, *Physical Review A* **99**, 10.1103/physreva.99.032331 (2019).
- [19] S. Efthymiou, S. Carrazza, A. Pasquale, M. Lazzarin, and A. Sopena, qiboteam/qibojit: qibojit 0.0.7 (2023).
- [20] S. Efthymiou *et al.*, qiboteam/qibo: Qibo 0.1.12 (2023).
- [21] S. Efthymiou *et al.*, qiboteam/qibolab: Qibolab 0.0.2 (2023).
- [22] A. Pasquale *et al.*, qiboteam/qibocal: Qibocal 0.0.1 (2023).

- [23] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, *Sympy: symbolic computing in python*, *PeerJ Computer Science* **3**, e103 (2017).
- [24] A. evolution in Qibo, <https://qibo.science/qibo/stable/code-examples/advancedexamples.html#trotterdecomp-example>.
- [25] J. Johansson, P. Nation, and F. Nori, *Qutip: An open-source python framework for the dynamics of open quantum systems*, *Computer Physics Communications* **183**, 1760–1772 (2012).
- [26] H. Chen and D. A. Lidar, *Hamiltonian open quantum system toolkit*, *Communications Physics* **5**, 10.1038/s42005-022-00887-2 (2022).
- [27] Z. Morrell, M. Vuffray, S. Misra, and C. Coffrin, *Quantumannealing: A julia package for simulating dynamics of transverse field ising models* (2024), arXiv:2404.14501 [quant-ph].
- [28] N. Hansen, *The CMA evolution strategy: A tutorial*, *CoRR abs/1604.00772* (2016), 1604.00772.
- [29] S. Blanes, F. Casas, J. Oteo, and J. Ros, *The magnus expansion and some of its applications*, *Physics Reports* **470**, 151–238 (2009).
- [30] S. Blanes, F. Casas, J. A. Oteo, and J. Ros, *A pedagogical approach to the magnus expansion*, *European Journal of Physics* **31**, 907 (2010).
- [31] J. Gonzalez-Conde, Z. Morrell, M. Vuffray, T. Albash, and C. Coffrin, *The Cost of Emulating a Small Quantum Annealing Problem in the Circuit-Model*, (2024), arXiv:2402.17667 [quant-ph].
- [32] A. Mari, T. R. Bromley, and N. Killoran, *Estimating the gradient and higher-order derivatives on quantum hardware*, *Physical Review A* **103**, 10.1103/physreva.103.012405 (2021).
- [33] S. Bertini, S. L. Cacciatori, and B. L. Cerchiai, *On the euler angles for SU(n)*, *Journal of Mathematical Physics* **47**, 043510 (2006).
- [34] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, *General parameter-shift rules for quantum gradients*, *Quantum* **6**, 677 (2022).
- [35] C. Bravo-Prieto, J. Baglio, M. Cè, A. Francis, D. M. Grabowska, and S. Carrazza, *Style-based quantum generative adversarial networks for monte carlo events*, *Quantum* **6**, 777 (2022).
- [36] A. Butter, T. Plehn, and R. Winterhalder, *How to GAN LHC Events*, *SciPost Phys.* **7**, 075 (2019), arXiv:1907.03764 [hep-ph].
- [37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, *Journal of Machine Learning Research* **12**, 2825 (2011).
- [38] J. Bergstra, B. Komer, C. Eliasmith, S. Cyphers, and D. Yamins, *Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms*, *Computational Science & Discovery* **8**, 014008 (2015).
- [39] Y. Y. Gao, M. A. Rol, S. Touzard, and C. Wang, *A practical guide for building superconducting quantum devices* (2021), arXiv:2106.06173 [quant-ph].
- [40] 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).
- [41] 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].
- [42] <https://github.com/qiboteam/qibo>.
- [43] <https://github.com/qiboteam/adiabatic-fit>.
- [44] M. Robbiati, J. Cruz-Martinez, and S. Carrazza, *adabatic-fit* (2023).