

# Determining probability density functions with adiabatic quantum computing

Matteo Robbiati,<sup>1,2</sup> Juan M. Cruz-Martinez,<sup>1</sup> and Stefano Carrazza<sup>1,2,3</sup>

<sup>1</sup>*CERN, Theoretical Physics Department, CH-1211 Geneva 23, Switzerland.*

<sup>2</sup>*TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano and INFN Sezione di Milano, Milan, Italy.*

<sup>3</sup>*Quantum Research Center, Technology Innovation Institute, Abu Dhabi, UAE.*

A reliable determination of probability density functions from data samples is still a relevant topic in scientific applications. In this work we investigate the possibility of defining an algorithm for density function estimation using adiabatic quantum computing. Starting from a sample of a one-dimensional distribution, we define a classical-to-quantum data embedding procedure which maps the empirical cumulative distribution function of the sample into time dependent Hamiltonian using adiabatic quantum evolution. The obtained Hamiltonian is then projected into a quantum circuit using the time evolution operator. Finally, the probability density function of the sample is obtained using quantum hardware differentiation through the parameter shift rule algorithm. We present successful numerical results for predefined known distributions and high-energy physics Monte Carlo simulation samples.

## I. INTRODUCTION

The determination of the underlying probability density function (PDF) of a given dataset is in general a challenging problem. In recent years several new approaches based on classic deep learning models have been proposed to tackle this fundamental problem. Some relevant examples are Variational Autoencoders [1, 2], Normalizing Flows [3], Riemann-Theta Boltzmann machines [4–6] and Generative Adversarial Networks [7].

Somewhat orthogonal to these developments, novel quantum inspired machine learning architectures have been introduced recently. The possibility to deploy successfully noisy intermediate-scale quantum (NISQ) computers [8] led to a growing interest in the development of a novel research field identified as Quantum Machine Learning (QML) [9]. Quantum neural networks (QNN) and parametrized quantum circuit [10–13], have been proposed for pattern classification [14–16], data compression [17, 18], data regression [19, 20] and generative models [21–23]. However, when considering the problem of PDF determination with QNN, even though models such as the Style-qGAN [23] can successfully generate samples, they cannot be utilized to determine a closed form expression for the underlying PDF. Furthermore, the training of a simple QNN to match the underlying PDF is not a simple and numerically stable option given the difficulty in defining boundaries, normalization and positivity constraints.

In this work we present a methodology which removes the training difficulties and constrains of QNNs by adopting an adiabatic quantum evolution strategy. In particular, we first define a regression model based on adiabatic evolution [24] which maps a generic one-dimensional function defined in a predefined bounded range as the time evolution of the expected energy of the adiabatic Hamiltonian as a function of time. This approach is sufficiently flexible to fit a large variety of functional forms and it can be

used to fit the empirical cumulative density function (CDF) as a monotonic increasing function bounded in the interval  $[0, 1]$ . After achieving an acceptable CDF regression, the method projects the obtained Hamiltonian in a quantum circuit representation using a Trotter-like decomposition [25] which predicts the trained function values. 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 empirical PDF of the sample as the derivative of the circuit using the Parameter Shift Rule (PSR) algorithm [26, 27].

The paper is organized as follows. In Sec. II we present the technical details of the probability density function estimation using adiabatic quantum computing. The Sec. III presents validation results for multiple examples. Finally, in Sec. IV 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 defined by two steps: the determination of an empirical cumulative distribution function using adiabatic quantum evolution as regression model, and subsequently, the determination of the probability density function through the trotter-like quantum circuit representation obtained from the adiabatic Hamiltonian.

### A. Model regression with adiabatic quantum evolution

Given a one-dimensional function,  $f(t)$ , we build a regression model by selecting an observable such that there are two Hamiltonians,  $H_0$  and  $H_1$  for which the respective energy ground states correspond to the two points between which we want to train the function. In this manuscript, for

simplicity, we set  $H_0 = X$  and  $H_1 = Z$  the non interacting Pauli matrices.

Therefore, we interpret the regression problem as the procedure of building a time dependent Hamiltonian  $H(t)$ , such that its ground state energy at each instant  $t$  is

$$\langle H(t) \rangle = f(t), \quad (1)$$

where  $f(t)$  is the target function usually defined in a bounded interval  $t \in [0, T]$ . Furthermore, using the quantum adiabatic evolution notation

$$H(t) = (1 - s(t; \theta))H_0 + s(t; \theta)H_1, \quad (2)$$

with  $s(t; \theta)$  the scheduling function which depends on a set of variational parameters  $\theta$ . The problem is then reduced to finding the right set of parameters  $\theta$  such that the adiabatic evolution of the state  $|\psi(t)\rangle$  from  $t = 0$  to  $t = T$  follows exactly the target function  $f(t)$ . Note that the choice of the functional form for the scheduling function  $s(t, \theta)$  is fundamental to guarantee flexibility or the monotonicity of the target function.

## B. Learning empirical cumulative density functions

The method presented above matches the requirements for a cumulative density function determination. The procedure follows a standard classical machine learning strategy: at first, we generate a sample of random variables  $\{x\}$  following a chosen distribution and we calculate the empirical CDF of the sample  $\{f\}$ . Then, we select  $N_{\text{train}}$  data elements such that their values match some of the evolution times controlled by the scheduling function. Each pair of  $(x_j, f_j)$  points are mapped into  $(\tau_j, E_j)$ , where  $\tau$  and  $E$  represent two generic values of the evolution time and the energy of our target observable evaluated at the evolved state at  $\tau = t/T$

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

$$J = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} (f_j - E_j(\theta))^2, \quad (3)$$

where  $E_j$  depends on the parameters through the scheduling function. In order to obtain a monotonic increasing regression function for the CDF determination, we can also add to  $J$  a penalty term for each pair of estimations  $f_j, f_{j+1}$  proportional to  $\Delta f$ . One can make the penalty term redundant by choosing a scheduling function which preserves monotonicity.

In Fig. 1 an example of this procedure is shown using quantum simulation on classical hardware with the Qibo framework [28–30]. Starting from a prior toy analytic CDF (black curve), data points are extracted in the interval  $x \in [0, 1]$ . The adiabatic regression model, defined

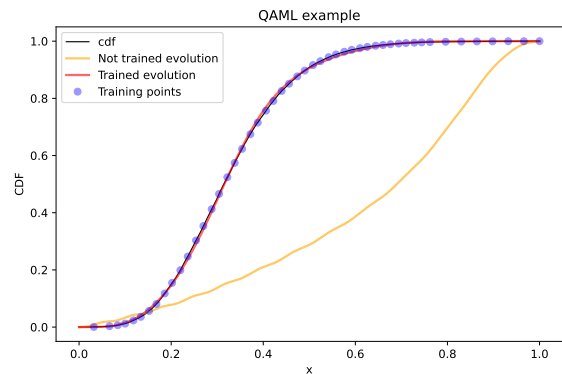


FIG. 1. Example of initial and final state of the algorithm. The  $N_{\text{train}}$  blue points are the training set selected from a gaussian mixture sample, whose empirical CDF is represented by the black line. The random initialization of the adiabatic evolution leads to the initial sequence of energies (yellow line). After a training time, the evolution is closer to the training set (red line).

in Sec. II A is drawn before training (yellow curve) and after a successful training (red curve). We perform the training of the model by optimizing the set of parameters involved into the scheduling  $s(t; \theta)$ , defined as the following polynomial of degree  $p$ :

$$s(t; \theta) = \frac{1}{\eta} \sum_{i=1}^p \theta_i x^i, \quad \text{with} \quad \eta = \sum_{i=1}^p \theta_i. \quad (4)$$

by using the CMA-ES optimizer [31]. In the example shown in (Fig. 1) we set  $p = 12$ , but the polynomial degree can be chosen depending on the complexity of the function one wants to fit. The Eq.(4) is an example of possible choices (one can also choose classical or quantum Neural Network); in particular, every scheduling function such that  $s(t = 0) = 0$  and  $s(t = 1) = 1$  can be used for performing an adiabatic evolution with Qibo. Note that other optimization algorithms may also work, including gradient-based strategies.

## C. Probability function from quantum circuits

### 1. A circuit for all times

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

by sequentially applying  $n$  operators:

$$|\psi(\tau_n)\rangle = \prod_{j=0}^n U(\tau_j) |\psi(\tau_0)\rangle = \mathcal{C}(\tau_n) |\psi(\tau_0)\rangle \quad (5)$$

where we write the product from  $n$  to zero to represent the order of application of the operators on the initial quantum state. Any sequence of unitary operator is itself an unitary operator, which we call  $\mathcal{C}_n = \mathcal{C}(\tau_n)$  and which evolves the initial state  $|\psi(\tau_0)\rangle$  to any point  $\tau_n = nd\tau$ .

In order to compute the state at any value of  $t$  outside of the discrete time steps of the adiabatic evolution  $\tau$  it is necessary to take the continuous limit. We start by considering one of the intermediate elements of the product:

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

where  $s_j$  is the value of the scheduling at evolution time  $\tau_j = jd\tau$ . This instantaneous form of the adiabatic hamiltonian operator can be diagonalized as  $\hat{D}_j$  using a matrix  $P_j$  such that:

$$\hat{H}_j = P_j \hat{D}_j P_j^{-1}, \quad (7)$$

with

$$P_j = \Lambda_j \begin{pmatrix} 1 & \frac{s_j - \lambda}{1 - s_j} \\ \frac{\lambda_j - s_j}{1 - s_j} & 1 \end{pmatrix}, \quad \hat{D}_j = \begin{pmatrix} \lambda_j & 0 \\ 0 & -\lambda_j \end{pmatrix}, \quad (8)$$

and  $\Lambda_j$  the appropriate ( $\tau$  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:

$$\mathcal{C}_n = \prod_{j=0}^n P_j e^{id\tau \hat{D}_j} P_j^{-1}. \quad (9)$$

If we now take the limit  $d\tau \rightarrow 0$ , we have that  $\hat{H}_j \rightarrow \hat{H}_{j-1}$  and thus  $P_j \rightarrow P_{j-1}$ . Thus adjacent elements in the sequence tend to the identity:  $P_j^{-1} P_{j-1} \rightarrow I$ . On this way, the Eq. (9) simplifies to:

$$\mathcal{C}_n = P_n \exp \left\{ i \sum_{j=0}^n \hat{D}_j d\tau \right\} P_0^{-1}. \quad (10)$$

Furthermore, in the limit of  $d\tau \rightarrow 0$  the sum in the above equation becomes an integration in  $d\tau$  with extremes  $\tau = 0$  and  $\tau = \tau_n$ . With the final expression for  $\mathcal{C}$  evaluated at any time  $t$  being:

$$\mathcal{C}(t) = P_t \exp \left\{ i \int_0^{t/T} \hat{D}_j d\tau \right\} P_0^{-1}, \quad (11)$$

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 time  $t$ .

## 2. Circuit representation

Let us implement the unitary operator  $\mathcal{C}(t)$ , which allows us to prepare a state in the ground state of  $H_0$  at  $t = 0$  and evolve it to any  $t$ , 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 rotations  $R_y$ ,  $R_z$ . Since any unitary operator  $U \in SU(2)$  can be written as a combination of three rotations [32] we choose:

$$\mathcal{U} \equiv R_z(\phi) R_x(\theta) R_z(\psi), \quad (12)$$

where the three angles ( $\phi, \theta, \psi$ ) can be computed as function of the matrix element of the operator  $\mathcal{C}$ :

$$\begin{cases} \phi = \pi/2 - \arg(c_{01}) - \arg(c_{00}) \\ \theta = -2 \arccos(|c_{00}|) \\ \psi = \arg(c_{01}) - \pi/2 - \arg(c_{00}). \end{cases} \quad (13)$$

The matrix elements  $c_{00}$  and  $c_{01}$  depend on the values of the scheduling  $s$  and the eigenvalues ( $\lambda$ ) of the Hamiltonian evaluated at a time  $t$ :

$$c_{0j} = \frac{1-s}{s\sqrt{\lambda(\lambda-s)}} \left\{ \cos I \left( 1 + (-1)^j \frac{\lambda-s}{1-s} \right) + i \sin I \left( 1 - (-1)^j \frac{\lambda-s}{1-s} \right) \right\}, \quad (14)$$

with  $I = \int_0^t \lambda(\tau) d\tau$ ,  $s = s(t)$  and  $\lambda = \lambda(t)$ .

Note that the construction of the circuit is completely independent of the choice of scheduling function  $s(t)$ .

## 3. From the CDF to the PDF

The circuit representation of the operator  $\mathcal{C}(t)$  allows us to reconstruct our original target function  $f(x)$  by applying the circuit to a state prepared at  $|\psi(0)\rangle$  and then measuring the desired observable, which in our case is a non-interacting Pauli  $\hat{Z}$ . Since with qibo we translate all the operations into a circuit representation, the expectation value of  $\hat{Z}$  is evaluated by executing the circuit  $N_{\text{shots}}$  times and then by calculating the probability of occurrence of the state  $|0\rangle$ . This expectation value is then used as estimation of the target function  $f(x)$ .

As previously said, our example case has been that in which the target function  $f(x)$  correspond to the empirical CDF of some arbitrary distribution.

By imposing monotonicity and pinning the initial and final points we ensure that its first derivative correspond to the PDF of the same distribution.

For a 1D distribution we have then:

$$\frac{df(t)}{dt} = \frac{d}{dt} \langle \psi_0 | \mathcal{C}(t)^\dagger \hat{Z} \mathcal{C}(t) | \psi_0 \rangle. \quad (15)$$

In the context of quantum computing, we can take advantage of what is usually known as Parameter Shift Rule (PSR) [26, 33] which allows us to take the derivative of an observable (such as Eq. (15)) by simply evaluating the circuit and shifting the parameters with respect to which we are taking the derivative. We are using specifically choice presented in Ref. [26] 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:

$$\text{PDF}(t) = \text{PSR} [\langle \psi_0 | \mathcal{C}(t)^\dagger \hat{Z} \mathcal{C}(t) | \psi_0 \rangle]. \quad (16)$$

In the following we “closure test” these techniques by drawing samples from a known distribution, building the circuit and reconstructing the original probability function.

### III. VALIDATION

#### A. Sampling known distributions

In order to validate and test the procedure, we select a number of known prior distributions. For each cases, we generate a representative sample of dimension  $N_{\text{sample}}$  and fit the resulting empirical CDF using the approach described in Sec. II A. We can then derive the PDF and compare results to the prior distribution. We repeat this exercise twice for every example by using quantum simulation on classical hardware with exact state-vector representation and with shots measurements.

In these examples the algorithm is set to stop once a given  $J_{\text{cut}}$  threshold of target precision is reached and the domain of the target variable is rescaled to be between 0 and 1 so that the interpretation of the observable as a CDF is direct. This is of course just a choice and the same techniques could be used to train functions defined in any arbitrary domain. In all cases the adiabatic evolution is run for  $T = 50\text{s}$  and with a time-step  $dt = 0.1$ . We define the scheduling function  $a$  as a polynomial with  $p$  parameters following the ansatz in Eq. 4.

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

$$\rho(x; \alpha, \beta) = \frac{\beta x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, \quad (17)$$

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

The results of the training can be seen in the first row and left column of Fig. 2 where we plot the true CDF together with both the exact and realistic simulation and

an histogram with the data used for the training. In the second row and left column of Fig. 2 we show the PDF obtained by taking the derivative of the circuit and compare to the original distribution from which the original points were sampled from. We also show the effect of modifying the number of shots (*i.e.*, the number of times we measure the qubit before accepting the results). In both these figures we show the exact results (*i.e.*, following the formula from Eq. (17)) in black and the exact simulation using state vectors in blue. In red instead, we show the results considering realistic circuits (where the collapse of the states is simulated by a classical sampling from the state vector). The error bands plots in the second row of Fig. 2 are computed by taking all realizations of the measurement and computing the standard deviation for each point in  $t$ . We plot two different choices ( $N_{\text{shots}} = \{2 \cdot 10^4, 2 \cdot 10^5\}$ ) for the number of shots in order to show how the result improves with the increased statistics.

In order to validate with a more complicated example we also sample from a Gaussian mixture defined as:

$$\rho(x; \vec{\mu}, \vec{\sigma}) = 0.6\mathcal{N}(x; \mu_1, \sigma_1) + 0.4\mathcal{N}(x; \mu_2, \sigma_2), \quad (18)$$

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.

In the right column of Fig. 2 we repeat the same study. The fit performs slightly worse due to the more complicated nature of the target function, the hardware already limits the level of precision that can be achieved.

In both these cases we train the adiabatic evolution as implemented in qibo [35–38]. While the parameter optimization is made by means of a CMA-ES [31] genetic algorithm. The final Hamiltonian is then transformed into a circuit as explained in section II C 1.

In order to produce the plots we perform two estimations. During simulation we can apply the circuit exactly to the ground state of  $H_0$  at every point in  $t$  such that we can show an “exact” (although ideal and nonphysical) situation. The more realistic result instead is estimated by running the same circuit several time simulating the actual randomness of a quantum device. This is implemented in Qibo through the method `AbstractHamiltonian.expectation_from_samples`, which can perform a realistic simulation.

When calculating the derivatives of the rotation angles with respect to  $t$ , which we call  $\partial_t \theta$ , some critical points of instability are found. For those  $t$ 's the value of the derivative of the angle increase exponentially; in the case of the exact simulation, the value of the PSR for the same points is very close to zero and this balances the divergence of  $\partial_t \theta$ . On contrary, when dealing with the realistic simulations, the PSR value is estimated through the mean of  $N_{\text{shots}}$  realizations and it can be possible that it is not close

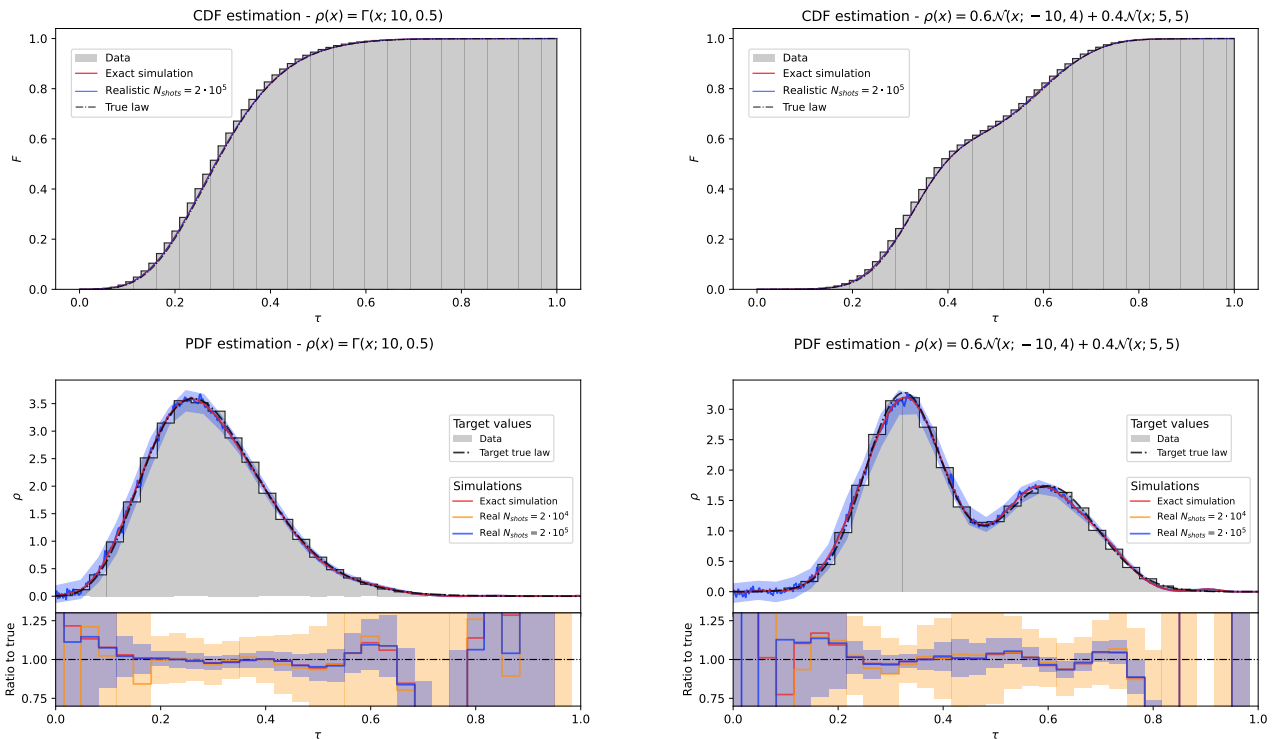


FIG. 2. Top row: comparisons between the true CDF and the result of training an adiabatic hamiltonian to follow the CDF from the sampled distribution. The exact CDF is represented by a discontinuous black line while the simulated circuit results are represented by a blue (realistic) and red (ideal) continuous lines. In grey an arbitrary binning of the data that has been used to train the circuit. Bottom row: comparison between the original PDF and the result of the derivative of the trained circuit. Once again the true result is represented by a discontinuous black, the data histogram is shown in grey and the exact simulation of the circuit is represented as a red continuous line. As regarding the realistic simulations, we show two different continuous lines, corresponding to different values of shots used for evaluating the expectation value of the target observable. In particular, we show in orange and blue the results obtained executing the circuit respectively  $N_{\text{shots}} = 2 \cdot 10^4$  and  $N_{\text{shots}} = 2 \cdot 10^5$  times. All of these results are shown in the form of a ratio between the target true law and the QAML estimations in the lowest part of the figures. While representing the PDFs, only one realistic simulation is drawn ( $N_{\text{shots}} = 2 \cdot 10^5$ ). All the realistic simulations curves are represented together with a  $1\sigma$  confidence belt calculated using  $N_{\text{runs}} = 20$  experiments.

to zero enough to balance the high values of the angles. In order to avoid a numerical instability for  $t_j$  (which leads to a few high peaks of the estimated PDFs), we smooth the realistic function by removing the outlier using the values registered for the neighbours of  $t_j$ .

### B. Density estimation using LHC data sampled from the Style-qGAN model

While in the previous case we were training the circuit using a sampling from a known distribution, we now move to the more complex case of learning an unknown distribution. We consider the particle physics process involving top and anti-top quark pair production ( $pp \rightarrow t\bar{t}$ ). While one could train directly by using the output of an event generator in our test case the data sampling is obtained from a separated circuit by using a style-based quantum

GAN (Style-qGAN) [39].

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 for the LHC configuration. For simplicity we limit the simulation at Leading Order in the strong coupling. The fit is done by training the model to the distributions of the rapidity  $y$  and the logarithms of the Mandelstam variables  $-\log(-t)$  and  $-\log s$ .

In Fig. 3 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 conventions as in the previous section.

In Table I we summarize the obtained results for all examples tested in this section. For each model we describe the final configuration and quality of the obtained model

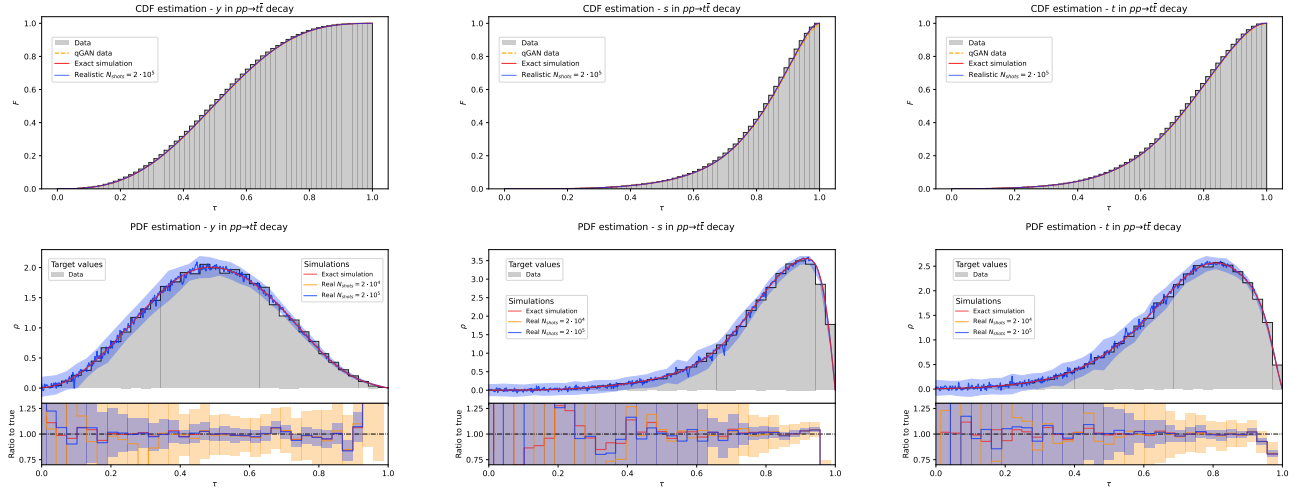


FIG. 3. The top row shows the CDF of the three HEP samples. In each figure, the histogram of the data is shown in gray, the CDF obtained by exact state vector simulation is shown in red, and the same result is shown in blue but considering realistic circuits executed  $N_{shots} = 2 \cdot 10^5$  times. The bottom row shows the PDF of three sampled HEP distributions. They are divided into two parts: above, in grey the histogram of the sample used to perform the fit in which we impose  $N_{mbins} = 34$  (this is the lowest number of bins thanks to which we can represent well the right side of the  $s$  and  $t$  distributions), and the curves representing the PDF of the sample: in red the law extracted via QAML using exact state vector simulation on  $N_{times} = 500$  times equally distributed between 0 and 1, and in blue the law obtained with the same approach but simulating a realistic circuit in which we calculate the expected value of the target observable by executing the circuit  $N_{shots} = 2 \cdot 10^5$  times. Below, we show the ratio between the histogram values of the PDF and our simulated results: the red line corresponds to the exact simulation, the orange and blue lines respectively correspond to realistic simulations in which we set  $N_{shots} = 2 \cdot 10^4$  and  $N_{shots} = 2 \cdot 10^5$ . All the realistic simulations curves are represented together with a  $1\sigma$  confidence belt calculated using  $N_{runs} = 20$  experiments.

Fit function	$N_{sample}$	$p$	$J_f$	$N_{ratio}$	$\chi^2$
Gamma	$5 \cdot 10^4$	25	$2.9 \cdot 10^{-6}$	31	$2.2 \cdot 10^{-4}$
Gaussian mix	$2 \cdot 10^5$	30	$2.75 \cdot 10^{-5}$	31	$4.39 \cdot 10^{-3}$
$t$	$5 \cdot 10^4$	20	$2.1 \cdot 10^{-6}$	34	$3.4 \cdot 10^{-4}$
$s$	$5 \cdot 10^4$	20	$7.9 \cdot 10^{-6}$	34	$1.20 \cdot 10^{-3}$
$y$	$5 \cdot 10^4$	8	$3.7 \cdot 10^{-6}$	34	$1.45 \cdot 10^{-3}$

TABLE I. Summary.  $N_{shots} = 5 \cdot 10^4$ .

by calculating the following test statistics:

$$\chi^2 = \sum_{i=0}^{N_{train}} \frac{(\hat{y}_i - y_i)^2}{y_i}. \quad (19)$$

In summary, the achieved level of quality is satisfactory for all tested distributions. We also observe that  $N_{shots} = 2 \cdot 10^5$  shots provides sufficient statistics to achieve accuracies into the range 4-10% for the ratio values (lower half of the PDFs plots). This accuracy range is obtained by calculating the ratio between the estimated sigma and the estimated PDFs ratio value for the points contained into a target range around the peak of the PDF; the minimum and the maximum values are used to define the range.

## IV. CONCLUSION

In this work we presented a methodology for the determination of probability density functions using adiabatic quantum computing. We first define a mechanism to use adiabatic evolution as a regression model for the fit of empirical cumulative density functions which are represented by the Trotterization of the adiabatic Hamiltonian in terms of a quantum circuit. The PDF is then obtained by applying the Parameter Shift Rule to the obtained circuit. This method allows the usage for training and inference of quantum devices designed for 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.

All numerical results have been obtained with Qibo framework [40], and are publicly available in [41]. Further possible developments include the generalization of this method for the simultaneous determination of multi-dimensional PDF distributions, the deployment of the full training procedure on quantum devices and the possibility to use quantum annealing for the optimization of the regression model parameters.

## ACKNOWLEDGMENTS

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

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

- 
- [1] D. P. Kingma and M. Welling, Auto-encoding variational bayes (2013).
- [2] D. J. Rezende, S. Mohamed, and D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models (2014).
- [3] D. J. Rezende and S. Mohamed, Variational inference with normalizing flows (2015).
- [4] D. Krefl, S. Carrazza, B. Haghghat, and J. Kahlen, *Neurocomputing* **388**, 334 (2020).
- [5] S. Carrazza and D. Krefl, *Computer Physics Communications* **256**, 107464 (2020).
- [6] A. Pasquale, D. Krefl, S. Carrazza, and F. Nielsen, Product jacobi-theta boltzmann machines with score matching (2023), arXiv:2303.05910 [stat.ML].
- [7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial networks (2014).
- [8] J. Preskill, *Quantum* **2**, 79 (2018).
- [9] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Nature* **549**, 195 (2017).
- [10] M. Larocca, N. Ju, D. García-Martín, P. J. Coles, and M. Cerezo, Theory of overparametrization in quantum neural networks (2021).
- [11] M. Schuld, R. Sweke, and J. J. Meyer, *Phys. Rev. A* **103**, 032430 (2021).
- [12] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, *Quantum Science and Technology* **4**, 043001 (2019).
- [13] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, *Advanced Quantum Technologies* **2**, 1900070 (2019).
- [14] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, *Nature* **567**, 209 (2019).
- [15] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, *Phys. Rev. A* **101**, 032308 (2020).
- [16] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, *Quantum* **4**, 226 (2020).
- [17] J. Romero, J. P. Olson, and A. Aspuru-Guzik, *Quantum Science and Technology* **2**, 045001 (2017).
- [18] A. Pepper, N. Tischler, and G. J. Pryde, *Phys. Rev. Lett.* **122**, 060501 (2019).
- [19] A. Pérez-Salinas, J. Cruz-Martinez, A. A. Alhajri, and S. Carrazza, *Phys. Rev. D* **103**, 034027 (2021).
- [20] 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].
- [21] P.-L. Dallaire-Demers and N. Killoran, *Phys. Rev. A* **98**, 012324 (2018).
- [22] S. Lloyd and C. Weedbrook, *Phys. Rev. Lett.* **121**, 040502 (2018).
- [23] C. Bravo-Prieto, J. Baglio, M. Cè, A. Francis, D. M. Grabowska, and S. Carrazza, *Quantum* **6**, 777 (2022).
- [24] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution (2000).
- [25] S. Paeckel, T. Köhler, A. Swoboda, S. R. Manmana, U. Schollwöck, and C. Hubig, *Annals of Physics* **411**, 167998 (2019).
- [26] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, *Physical Review A* **98**, 10.1103/physreva.98.032309 (2018).
- [27] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, *Phys. Rev. A* **99**, 032331 (2019).
- [28] 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).
- [29] S. Efthymiou, M. Lazzarin, A. Pasquale, and S. Carrazza, *Quantum* **6**, 814 (2022).
- [30] S. Carrazza, S. Efthymiou, M. Lazzarin, and A. Pasquale, *Journal of Physics: Conference Series* **2438**, 012148 (2023).
- [31] N. Hansen, CoRR **abs/1604.00772** (2016), 1604.00772.
- [32] S. Bertini, S. L. Cacciatori, and B. L. Cerchiai, *Journal of Mathematical Physics* **47**, 043510 (2006).
- [33] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, *Physical Review A* **99**, 10.1103/physreva.99.032331 (2019).
- [34] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, *Quantum* **6**, 677 (2022).
- [35] S. Efthymiou, S. Carrazza, A. Pasquale, M. Lazzarin, and A. Sopena, qiboteam/qibojit: qibojit 0.0.7 (2023).
- [36] The Qibo Team, qiboteam/qibo: Qibo 0.1.12 (2023).
- [37] The Qibo Team, qiboteam/qibolab: Qibolab 0.0.2 (2023).
- [38] The Qibo Team, qiboteam/qibocal: Qibocal 0.0.1 (2023).
- [39] C. Bravo-Prieto, J. Baglio, M. Cè, A. Francis, D. M. Grabowska, and S. Carrazza, *Quantum* **6**, 777 (2022).
- [40] <https://github.com/qiboteam/qibo>.
- [41] <https://github.com/qiboteam/adiabatic-fit>.