PAPER • OPEN ACCESS

Multi-variable integration with a variational quantum circuit

To cite this article: Juan M Cruz-Martinez *et al* 2024 *Quantum Sci. Technol.* **9** 035053

View the article online for updates and enhancements.

Quantum Science and Technology

Multi-variable integration with a variational quantum circuit

Juan M Cruz-Martinez[1,*] , Matteo Robbiati[1,2] and Stefano Carrazza[1,2,3]

1   Theoretical Physics Department, CERN, CH-1211 Geneva 23, Switzerland
2   TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano and INFN Sezione di Milano, Milan, Italy
3   Quantum Research Center, Technology Innovation Institute, Abu Dhabi, United Arab Emirates
*   Author to whom any correspondence should be addressed.

E-mail: juan.cruz.martinez@cern.ch

Keywords: quantum machine learning, integration methods, high energy physics

Abstract

In this work we present a novel strategy to evaluate multi-variable integrals with quantum circuits. The procedure first encodes the integration variables into a parametric circuit. The obtained circuit is then derived with respect to the integration variables using the parameter shift rule technique. The observable representing the derivative is then used as the predictor of the target integrand function following a quantum machine learning approach. The integral is then estimated using the fundamental theorem of integral calculus by evaluating the original circuit. Embedding data according to a reuploading strategy, multi-dimensional variables can be easily encoded into the circuit's gates and then individually taken as targets while deriving the circuit. These techniques can be exploited to partially integrate a function or to quickly compute parametric integrands within the training hyperspace.

1. Introduction

Many scientific and engineering problems require the evaluation of numerical integrals of varying complexity of the form:

$$I(\boldsymbol{\alpha}) = \int_{\boldsymbol{x}_a}^{\boldsymbol{x}_b} g(\boldsymbol{\alpha}; \boldsymbol{x})\, \mathrm{d}^n \boldsymbol{x}, \tag{1}$$

where the bold symbols correspond to vectors, $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ define the integration domain and $g(\boldsymbol{\alpha}; \boldsymbol{x})$ is an integrand which depends on the integration variables $(\boldsymbol{x})$ and some parameters $(\boldsymbol{\alpha})$.

There are many numerical integration methods which tackle this problem, and the choice usually depends on the characteristics of the integrand functions and the integral region. For instance, low-dimensional well-behaved integrals can be successfully integrated with quadrature methods. However, achieving the same precision with more complicated or higher-dimensional integrands will lead to a significant increase in computational costs.

In those cases, Monte Carlo (MC) methods are often favored due to their ability to handle a wider range of integrand functions without imposing stringent requirements and a convergence rate that does not depend on the dimensionality of the integrand. An important feature of MC methods is the possibility of binning partial results so that differential distributions in any of the integration variables can be obtained. However, in exchange for their flexibility, they suffer from slow convergence and require a large number of function evaluations. To mitigate these issues, various techniques have been proposed to speed up the integration process, by reducing the number of integrand evaluations while producing accurate results [1–5]. In the context of particle physics, these advantages have made the VEGAS [6, 7] MC algorithm into the gold standard for numerical integration. Over the past decade, there have been numerous attempts to further improve the algorithm without modifying the underlying strategy. Some examples are the implementation of the algorithm in new hardware devices [8, 9], which offer a raw speed-up over traditional computing; the usage of multi-channel techniques [10], which exploit prior knowledge of the behavior of the different pieces

of the integrand; or machine learning (ML) techniques to enhance the importance sampling algorithm [11, 12].

However, all these methods suffer from the same drawback: obtaining a result requires the repeated numerical evaluation of the integrand in the region of interest. In addition, even if the integral smoothly depends on parameters which are not integrated over (like α in equation (1)), any change in them requires a complete new run.

In references [13, 14] a new approach has been proposed which can be utilized to circumvent this issue. These methods are based on using artificial neural networks (aNN) to build a surrogate model for the primitive of the integrand. Such a surrogate can keep the dependence on both the integration variables and the function parameters within the integration domain in which it has been trained. By subsequently training the derivative of the model to approximate the integrand it is possible to recover its primitive. While the computational cost does not disappear (it is translated to the training process), it allows for the evaluation of the integral (or the marginalization of the integrand over any of the variables) for any choice of parameters within the training range at almost zero additional cost.

In this paper we apply the same strategy in a quantum machine learning (QML) [15–19] context to estimate the value of integrals of the form of equation (1). The unique aspects of quantum circuits renders them an exceptional tool for this methodology. In the classical version, we need to train the derivative of the aNN. By taking the derivative, the architecture of the network can considerably change, possibly leading to a costly hyperparameter search in order to find the optimal model [14]. With a quantum circuit instead, we can exploit their properties in order to obtain the derivative using the parameter shift rule (PSR) [20–23], therefore using the same architecture for the derivative and its primitive.

For doing this, we use `Qibo` [24–30], a full-stack and open-source framework for quantum simulation, control and calibration.

The rapid development of quantum technologies leads us to believe that quantum circuits and QML tools can be exploited to improve on the performance, despite the constraints imposed by the current era of limited near-intermediate scale quantum [31] (NISQ) devices. In particular, we note quite some interest on the field of high energy physics where many new algorithms are being developed and tested leading to a very robust ecosystem of quantum computing tools focusing on particle physics [32–39]

This paper is structured as follows, we expose the method in section 2, after a brief introduction to QML and circuits derivative calculation respectively in Section 2.1 and 2.3. In section 3 we apply the method to two situations, a toy-model represented by a d-dimensional trigonometric function and a real-life scenario motivated by particle physics.

All results can be reproduced using the code at: https://github.com/qiboteam/QiNNtegrate.

2. Methodology

In this section we introduce well known concepts of quantum computation (QC) which are useful to better understand the methodology. Then, we describe the integration procedure more in detail.

2.1. QML in a nutshell

Classical ML is nowadays widely used to tackle statistical problems, such as classification, regression, density estimation, pattern recognition, etc. The goal of the ML algorithms is to teach a model to perform some specific task through an iterative optimization process. Let us recall here some basic ML concepts which equally apply to QML in order to establish the notation used through the paper. For simplicity, we treat here the case of Supervised ML, but what follows can be easily extended to other approaches.

We consider two variables: an input vector $\boldsymbol{x}$ and an output vector $\boldsymbol{y}$, which is related to $\boldsymbol{x}$ through some hidden law

$$\boldsymbol{y} = f(\boldsymbol{x}), \tag{2}$$

which we aim to estimate. These vectors can be composed of any number of variables and the dimensionality of the input and output data can in general be different. A model $\mathcal{M}_{\boldsymbol{\theta}}$, parametric in some parameters $\boldsymbol{\theta}$, is then chosen to make predictions $\boldsymbol{y}_{\text{est}}(\boldsymbol{x}|\boldsymbol{\theta}) = \mathcal{M}_{\boldsymbol{\theta}}(\boldsymbol{x})$ of the output variables. Once a model is selected, a loss function J is typically used to verify the predictive goodness of $\mathcal{M}_{\boldsymbol{\theta}}$. Finally, an optimizer is required, which is in charge of estimating:

$$\boldsymbol{\theta}_{\text{best}} = \operatorname{argmin}_{\boldsymbol{\theta}} \left\{ \sum_i J\left[\boldsymbol{y}_{i,\text{est}}(\boldsymbol{x}_i|\boldsymbol{\theta}), \boldsymbol{y}_{i,\text{meas}} \right] \right\}. \tag{3}$$

where $\boldsymbol{y}_{\text{meas}}$ is a measured realization of $\boldsymbol{y}$.

In QC we use two-level quantum systems called *qubits* to store information. One of the most popular quantum computing paradigms is the *Gate based Quantum Computing*, where the qubits state is manipulated with the action of unitary operators we call *gates*, defining a totally reversible computation. These unitaries can be combined to build up multi-qubit gates. In practice, QC introduces new computational possibilities with respect to the classical counterpart with tools such as superposition and entanglement.

In the context of QML, a common approach is to use variational quantum circuits (VQC) [18, 40, 41] as a quantum version of the classical parametric models. A VQC is a collection of gates which depends on a set of parameters $\boldsymbol{\theta}$, which can be used to regulate the manipulation of a quantum state. A circuit can be applied to an initial quantum state $|\psi_i\rangle$ and, after the execution, measurements can be performed on the final state $|\psi_f\rangle$. We can collect information by executing the circuit N_{nshots} times, and then calculating expected values of the target observables $\hat{O}$,

$$G_{\hat{O}}(\boldsymbol{\theta}) = \langle\psi_i|\mathcal{C}^\dagger(\boldsymbol{\theta})\,\hat{O}\mathcal{C}(\boldsymbol{\theta})\,|\psi_i\rangle. \tag{4}$$

In the following, for simplicity, we omit the reference to the observable, which is chosen to be a non-interacting Pauli $\hat{\sigma}_z$ taken independently over all qubits and then averaged over the number of qubits.

The expectation values introduced in equation (4) can be used as predictions $\boldsymbol{y}_{\text{est}} = G(\boldsymbol{\theta})$ in the QML process.

Several methods are known to embed data into a quantum system [42–45]. In this work we follow the procedure presented in [43] and known as *data re-uploading*, which allows us to encode multi-dimensional variables into the parametric gates of a circuit as part of their parameters. Since we encode the data directly into the gates of the VQC, we do not need any additional state preparation and from now on we consider $|\psi_i\rangle = |0\rangle$.

Adopting this embedding strategy, the final predictions will be obtained executing a circuit which is explicitly depending on the data:

$$\boldsymbol{y}_{\text{est}} = G(\boldsymbol{x}|\boldsymbol{\theta}) = \langle0|\mathcal{C}^\dagger(\boldsymbol{x}|\boldsymbol{\theta})\,\hat{O}\mathcal{C}(\boldsymbol{x}|\boldsymbol{\theta})\,|0\rangle. \tag{5}$$

Finally, there are numerous possible choices for the loss function and the optimizer, depending on the type of model chosen and whether one is doing simulation or executing on a real quantum hardware. For example, by doing simulation of a VQC, a natural choice is to select a well known gradient-based optimizer [5, 46–50] or some meta-heuristic algorithm like evolutionary strategies [51], simulated annealing [52], etc. Instead, when deploying the algorithm in actual quantum hardware it can be more effective to use shot-frugal optimizers [53–55] or to calculate gradients using metrics better suited to the QC context [56].

2.2. Circuit's ansatz

Building up our QML models, we encode the input data into the parametric gates of the circuit following the strategy suggested in [43], according to which an external variable can be uploaded into the angle ϕ of rotational gates of the form:

$$R_k(\phi) = \exp{-i\phi\hat{\sigma}_k}, \tag{6}$$

where the hermitian generator of the rotation $\hat{\sigma}_k$ is one of the Pauli's matrices.

In particular, we implement an architecture inspired by the uploading layer described in [43] called *fundamental Fourier Gate*, $\mathcal{U}$, which is composed of five sequential rotations around the z and the y axis. Our $\mathcal{U}$ implementation is as follows:

$$\mathcal{U}(x|\boldsymbol{\theta}) = R_z(\theta_1)\,R_y(\theta_2)\,R_z(\theta_3)\,R_z(\theta_4 x)\,R_y(\theta_5), \tag{7}$$

where the data x is uploaded into the second gate in order of application on the initial state. The power of this approach lies in the fact that by re-uploading the data x into N consecutive channels in the form of equation (7), we approximate a target continuous function as would an N-term Fourier series [57].

This strategy is introduced for a single qubit system in which a single variable is re-uploaded, but is easily extendible to a many-qubit case. Moreover, increasing the number of qubits also increases the flexibility of the model, allowing us to upload different variables into different wires of the circuit. Several choices of architecture can be done, and we present here two of the various models implemented within the code accompanying this work.

The first one is shown in figure (1), we encode two dimensions in every qubit, such that the width of the circuit is equal to half the number of dimensions ($N_{\text{dim}}/2$). Each uploading of the couple of variables (x_j, x_{j+1}) into the associated qubit is in the form presented in equation (7).
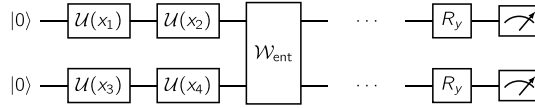
Figure 1. Diagram representation of the reuploading ansatz used to fit a 4-dim function. The $\mathcal{U}(x_i)$ quantum channel corresponds to the fundamental Fourier Gate presented in [43], while the entangling channel $\mathcal{W}_{ent}$ is built with a combination of *CZ* gates. This encoding layer is repeated N_{layers} times. Finally, an R_y gate is added to each qubit. All parameters of the ansatz are included in the training.
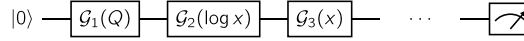
Figure 2. Schematic representation of the qPDF ansatz used to fit the *u* quark PDF. The presented gates have to be considered as a single layer of the ansatz, which is then repeated N_{layers} times.

Each family of gates $\{\mathcal{U}_j, \mathcal{U}_{j+1}\}$, with $\mathcal{U}_j = \mathcal{U}(x_j)$, is then followed by an entangling channel $\mathcal{W}_{ent}$, which distributes the information accumulated by each qubit to the entire system. After the last layer, a final rotation R_y is added to each wire of the circuit before performing the measurements.

Secondly, we implement a two-dimensional extension of the qPDF model from reference [58]. This second ansatz, is shown in figure (2), where we define the following channels:

$$\begin{cases} \mathcal{G}_1(Q) = R_y(\alpha_1 Q + \beta_1), \\ \mathcal{G}_2(\log x) = R_z(\alpha_2 \log x + \beta_2), \\ \mathcal{G}_3(x) = R_y(\alpha_3 x + \beta_3), \end{cases} \tag{8}$$

with α_i and β_i variational parameters. The input variables x and Q represent, respectively, a momentum fraction and an energy, and will be more thoroughly described in the sequel.

The qPDF model is introduced in reference [58] to fit a parton distribution function (PDF) using a VQC. PDFs are quasi probability distributions describing the partonic (gluons and quarks) content of the proton in hadron collisions. For the purposes of this study, it is enough to describe them as the probability of finding a given parton (for instance, the *u*-quark) at an energy Q carrying a fraction of the total momentum of the proton x. PDFs can only be computed from first principles in specific limits and thus the usual strategy is to obtain the distribution as a fit to data. In reference [58] the NNPDF framework [59] is extended from using a NN to use a VQC instead. Other approaches include Gaussian Processes [60] or polynomial forms. For a more detailed discussion on PDFs and the techniques used in their determination, we refer the reader to reference [61].

In equation (8) the imbalance of 2:1 in uploading the x and Q variables into the model follows the strategy of reference [62] to capture the different behaviors of the PDF at low values of x (logarithmic) and high values (linear).

This is one example of how a circuit designed to approximate a family of problems (in this case PDFs) can also be satisfactory exploited to integrate said family of functions thanks to the parameter shift rule, which will be briefly described in the next section.

2.3. Derivative of a quantum circuit

Our aim is then to use the derivative of G defined in equation (4) with respect to $\boldsymbol{x}$ as predictor of the integrand function presented in equation (1). For this we use the PSR as the method for calculating the derivatives of G with respect to $\boldsymbol{x}$. The first example of PSR was presented in [17] and introduced a method for calculating the derivative of an expectation value in the form of equation (4) with respect to one of the rotation angles affecting the quantum circuit $\mathcal{C}$. We refer to a more general PSR formula presented in [20] and further developed later [21–23, 63]. According to [20], if a circuit depends on a parameter μ (with μ a component of the vector $\boldsymbol{\theta}$) through a single gate U whose hermitian generator has at most two eigenvalues, the derivative of G with respect to μ can be exactly calculated as follows:

$$g(\mu) \equiv \partial_\mu G(\boldsymbol{\theta}) = r\left(G(\mu^+) - G(\mu^-)\right), \tag{9}$$

where $\pm r$ are the eigenvalues of the generator of U, $\mu^\pm = \mu \pm s$ and $s = \pi/4r$. In this work we limit ourselves to rotational gates such as equation (6) for which $r = \frac{1}{2}$ and $s = \frac{\pi}{2}$. The derivative of the circuit corresponds then to the execution of the same circuit twice per gate to which the input parameter has been uploaded to.

Since we never upload two input parameters to the same gate, the multidimensional extension is a trivial sequential application of the PSR per dimension and gate. If every dimension is uploaded once to every layer (l), the total number of expectation values necessary is $(2l)^{N_{\dim}}$. Note that the number of input parameters does not need to coincide with the dimensionality of the integral, making this method particularly useful for parametric integrals which are less prone to the so-called curse of dimensionality.

2.4. Solving integrals with quantum circuits

In the following section we describe the QML training procedure and, once the optimization is done, how the final model can be used to calculate the integral of the target function.

Calculating the derivative of the circuit with the PSR, we are able to use the same architecture to evaluate the primitive of a function G and any of its derivatives g. To be more explicit, if we recall the formula with which we started the paper:

$$I(\boldsymbol{\alpha}) = \int_{\boldsymbol{x}_a}^{\boldsymbol{x}_b} g(\boldsymbol{\alpha};\boldsymbol{x})\,\mathrm{d}^n\boldsymbol{x}, \tag{10}$$

the finite integral $I(\boldsymbol{\alpha})$ can also be calculated in the hypercube defined by $(\boldsymbol{x}_a, \boldsymbol{x}_b)$ by using its primitive $G(\boldsymbol{x};\boldsymbol{\alpha})$:

$$I(\boldsymbol{\alpha}) = \sum_{x_1,\cdots,x_n = x_a,x_b} (-1)^{\#a}\, G(x_1,\cdots,x_n;\boldsymbol{\alpha}), \tag{11}$$

where the sum runs over all combinations of the integration limits a and b and $\#a$ counts the number of variables evaluated in the lower limit a. In the one-dimensional case the equation above simplifies to:

$$I(\boldsymbol{\alpha}) = G(x_b;\boldsymbol{\alpha}) - G(x_a;\boldsymbol{\alpha}), \tag{12}$$

with

$$G(x;\boldsymbol{\alpha}) = \int g(\boldsymbol{\alpha};x)\,\mathrm{d}x. \tag{13}$$

Our goal will be training the derivative of a VQC such that it approximates the function g at any point $\boldsymbol{x}_j$ within the integration limits $(\boldsymbol{x}_a, \boldsymbol{x}_b)$.

$$g_{j,\mathrm{est}}(\boldsymbol{\alpha};\boldsymbol{x}_j|\boldsymbol{\theta}) = \left.\frac{\partial G(\boldsymbol{\alpha},x_1,\ldots,x_n|\boldsymbol{\theta})}{\partial x_1\ldots\partial x_n}\right|_{\boldsymbol{x}_j}. \tag{14}$$

If we have a way of evaluating $g(\boldsymbol{\alpha};\boldsymbol{x})$ or have access to measurements of its value we can generate a set of N_{train} training data. Once the predictions $\{g_{j,\mathrm{est}}\}_{j=1}^{N_{\mathrm{train}}}$ are calculated for all the training data, we quantify the goodness of our model by evaluating a Mean-Squared Error loss function:

$$J_{\mathrm{mse}} = \frac{1}{N_{\mathrm{train}}} \sum_{j=1}^{N_{\mathrm{train}}} \left[g_{j,\mathrm{meas}} - g_{j,\mathrm{est}}(\boldsymbol{\alpha};\boldsymbol{x}_j|\boldsymbol{\theta})\right]^2, \tag{15}$$

where we indicate with the index (j) the jth element in the training dataset and its associated integrand value. The training is thus performed by iteratively updating the parameters $\boldsymbol{\theta}$ in order to minimize equation (15). Various optimizers have been tested, including Powell method [64], L-BFGS [65], a covariance matrix adaptation evolutionary strategy [51] (CMA-ES) and a Basin-Hopping algorithm [66]. Even though benchmarking different optimizers performances goes beyond the scope of this paper, we stress the importance of considering different optimization approaches when training variational quantum algorithms with and without shot-noise, namely, simulating the sampling of the final state in the selected measurement basis. In fact, some of the optimizers which lead to very good results in the case of exact simulation (e.g. L-BFGS), become unusable when activating the shot-noise. This is expected, since such algorithms make use of numerical differentiation during the optimization process, and the numerical gradients cannot be computed in a loss function landscape which suffers from the natural randomness provoked by the shot-noise. To implement gradient-based optimization strategies which are compatible with the shot-noise computation, one should execute the calculation of the gradients using techniques which are more robust to the randomness of the system, such as parameter-shift rules [20]. Although this can be a perfectly viable choice, we decide to follow different strategies in this work, not to worry about scalability problems like those introduced in [67].
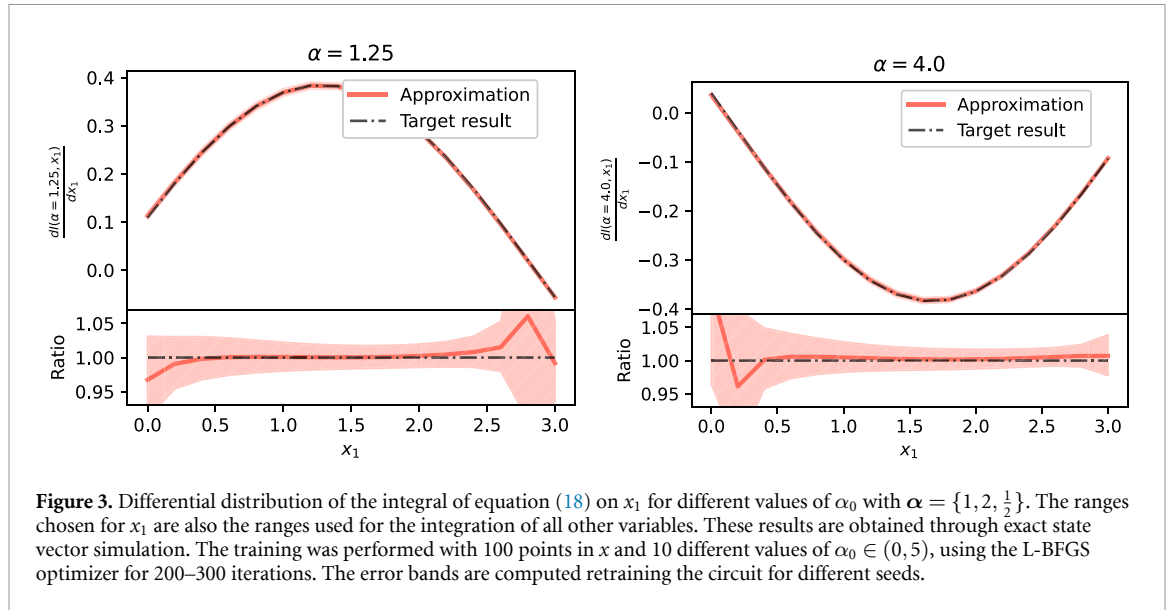
Figure 3. Differential distribution of the integral of equation (18) on x_1 for different values of α_0 with $\boldsymbol{\alpha} = \{1, 2, \frac{1}{2}\}$. The ranges chosen for x_1 are also the ranges used for the integration of all other variables. These results are obtained through exact state vector simulation. The training was performed with 100 points in x and 10 different values of $\alpha_0 \in (0, 5)$, using the L-BFGS optimizer for 200–300 iterations. The error bands are computed retraining the circuit for different seeds.

We adopt two different optimization approaches in the two training scenarios, using L-BFGS when optimizing in the exact simulation regime, and preferring heuristic algorithms like CMA-ES and Basin-Hopping when activating the shot-noise simulation.

Once the circuit's parameters are optimized, we have a fixed architecture which is a surrogate of $G(x; \boldsymbol{\alpha})$ and that can be used to evaluate integrals with respect to any combination of the target variables. This aspect makes the strategy particularly interesting when dealing with high-dimensional functions.

As an example, we can marginalize the integrand previously defined over the variable x_k

$$I_{ab}(\ldots, x_{k-1}, x_{k+1}, \ldots) = \int_{x_{k,a}}^{x_{k,b}} g(\boldsymbol{x}) \, \mathrm{d}x_k, \tag{16}$$

by uploading the integration limits, $x_{k,a}$ and $x_{k,b}$ and removing only that derivative from equation (14):

$$I_{ab}(\ldots, x_{k-1}, x_{k+1}, \ldots) \simeq g_{\mathrm{est}}(x_{k,b}|\boldsymbol{\theta}_{\mathrm{best}}) - g_{\mathrm{est}}(x_{k,a}|\boldsymbol{\theta}_{\mathrm{best}}), \tag{17}$$

where we write g_{est} explicitly depending only on x_k for simplicity. This can be extended for the partial integration of any of the variables of which I depends on, until finally we recover the full integration as shown in equation (11).

3. Results

In order to showcase the possibilities of the methodology presented in this paper we are going to use a VQC for two different target functions. We will first show the flexibility of the method to obtain total or partial integrals and differential distributions, and then we will apply to a practical case in which the approach can introduce a net-gain. Both examples are implemented in the public code which accompanies this paper (among other examples).

3.1. Toy model
For our first example we utilize the ansatz of reference 1 to approximate a d-dimensional trigonometric function:

$$g(\boldsymbol{x}) = \cos(\boldsymbol{\alpha} \cdot \boldsymbol{x} + \alpha_0), \tag{18}$$

with $\boldsymbol{x}$ and $\boldsymbol{\alpha}$ n-dim vectors. The integral of equation (18), while trivial to perform analytically, will serve to demonstrate how training one single circuit to obtain the primitive,

$$I(\boldsymbol{\alpha}; \boldsymbol{x}) = \int g(\boldsymbol{\alpha}; \boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{19}$$

can provide us the flexibility to obtain other derived quantities.

For instance, we might be interested on the differential distributions $\frac{\mathrm{d}I(\boldsymbol{\alpha};\boldsymbol{x})}{\mathrm{d}x_i}$ for a given i and for different values of one of the parameters $\boldsymbol{\alpha}$. In general, this would require to perform the numerical integration once per choice of i, per bin in the distribution and choice of α. By having a surrogate for $I(\boldsymbol{\alpha};\boldsymbol{x})$ we can obtain each distribution as seen in figure 3, were we collect results obtained by training the model with exact state vector simulation of the quantum circuits.

In figure 3 we have plotted the differential distribution

$$\frac{\mathrm{d}I(\boldsymbol{\alpha};x_1)}{\mathrm{d}x_1}, \tag{20}$$

for two different values of α_0 within the training range $(0,5)$. All other parameters have remained fixed in order to minimize the computing cost in this toy-model example. The training range for the integrated variables (x_1, x_2, x_3) has been $(0, 3.5)$. In the plots we choose to integrate x_2 and x_3 from 0 to 3 for every value fo x_1, but any choice of integration limits within the integration range would be possible.

A shortcoming of this approach is the lack of an uncertainty associated to the numerical integration. In reference [14] the suggestion is to use an ensemble of replicas of the network trained to the same data in order to use the variance as an error. We have followed the same strategy here by training an ensemble of circuits randomizing the choice of training points which leads to a spread of the results.

Other numerical methods provide some shortcuts to obtain similar results. For instance MC integration methods would allow us to bin quantities which depend on the integration variables (provided that we know beforehand the distributions that we want to obtain). However, a change in the parameter α_0 will always lead to a new integration. Instead, once we have a circuit that approximates equation (19), any derived quantity in the training range is accessible without any new runs.

It is important to remark that there is no free lunch, we are paying the penalty in terms of evaluations of the integrand (and the surrogate) during the training, but the outcome is a flexible representation of the final quantity which can then be reutilized. Similarly to MC methods, the accuracy of the calculation can be improved by increasing computational cost with either a larger number of samples or longer training lengths. Note that in this case we have a function that enables us to generate an unlimited amount of data for arbitrary inputs and so the limitation is only of computational cost.

3.2. The *u*-quark PDF

In the previous section we have chosen an ideal scenario with a function for which we know the primitive and against which we can exactly test. In what follows we consider an actual use-case for the approach that we propose in this paper: the integration of a function which is only known numerically and for which we have a representation in the form of a VQC.

For this we use the PDF fit and corresponding ansatz presented in reference [58]. Note that in order to overcome some of the computational challenges that arose when swapping the NN for a VQC, in reference [58] the resulting PDF is not normalized. Normalizing the PDF requires taking the integral over x at the fixed fitting energy (Q_0), which in turn requires many evaluations of the PDF at every stage of the fit. Instead, with the techniques introduced in this paper we use the ansatz described in section 2.2 to obtain a model by which we can produce both the *u*-quark PDF and its own integral, hence allowing for a prediction normalized by construction without the need for an expensive numerical integral.

In figure 4 we show the result of fitting the derivative of the ansatz to the training data for a fixed value of Q, obtaining a very good description across the entire range.

We train the model with the L-BFGS optimizer and performing exact simulation using `Qibo`. As the training data we utilize directly the *u*-quark PDF from the NNPDF4.0 [62] PDF grids. We limit the training to the x range $(1 \times 10^{-4}, 0.7)$ which is comparable with the ranges of data available in PDF determinations. In figure 4 we train for several fixed values of Q chosen linearly between 1.65 and 40 GeV. Once we are able to approximate the integrand with the derivative of the ansatz, the associated integral,

$$I_u(Q) = \int_{10^{-4}}^{0.7} x u(x, Q) \, \mathrm{d}x, \tag{21}$$

is the ingredient necessary for the normalization of the *u*-quark PDF.

Note that the Q-dependence of the PDF can be computed analytically given the PDF at a fixed scale Q_0, thus in a fit only a value of $I_u(Q_0)$ is required. The ability to train the VQC for a range of values of Q can be exploited to evaluate equation (21) as a parametric integral for a range of values of Q.
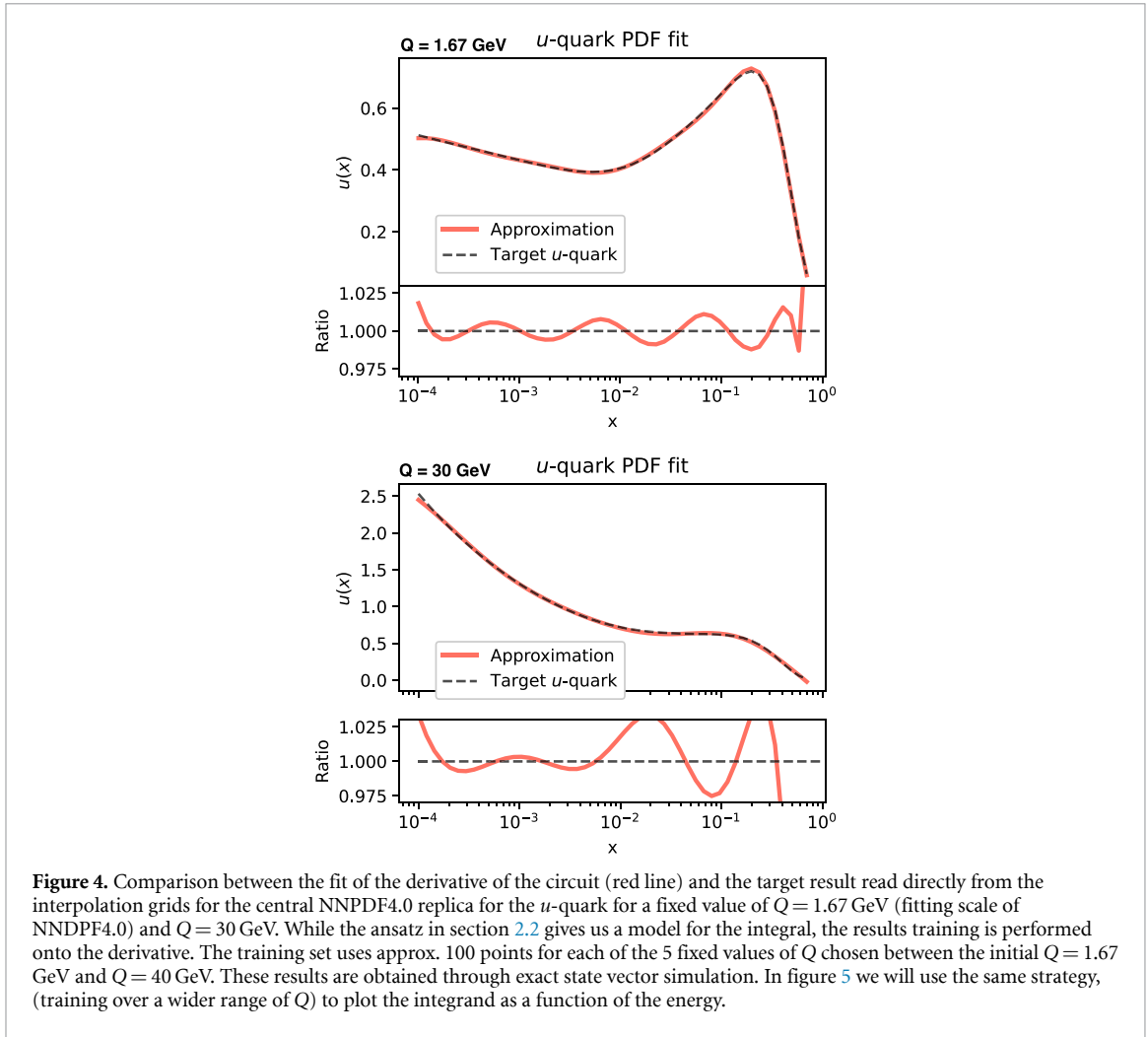
Figure 4. Comparison between the fit of the derivative of the circuit (red line) and the target result read directly from the interpolation grids for the central NNPDF4.0 replica for the *u*-quark for a fixed value of $Q = 1.67$ GeV (fitting scale of NNDPF4.0) and $Q = 30$ GeV. While the ansatz in section 2.2 gives us a model for the integral, the results training is performed onto the derivative. The training set uses approx. 100 points for each of the 5 fixed values of Q chosen between the initial $Q = 1.67$ GeV and $Q = 40$ GeV. These results are obtained through exact state vector simulation. In figure 5 we will use the same strategy, (training over a wider range of Q) to plot the integrand as a function of the energy.

In order to show the generalizability potential of the method we have also trained the circuit for a wide range of values of Q, chosen such that no quark-threshold is crossed (defined as the value of the energy for which the number of active quarks in NNPDF4.0 changes) to reduce instabilities.
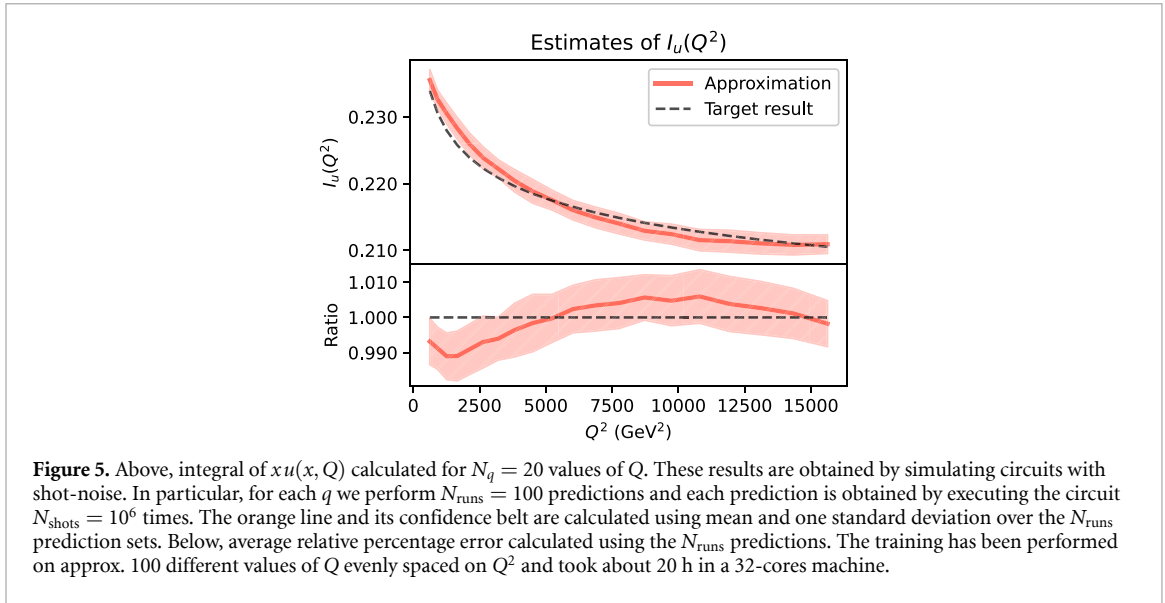
In QML, in real-life scenarios, one needs to account for the probabilistic shot-noise associated with the measurement of the quantum states and the noise associated to the hardware. While the uncertainty associated to the training shown in figure 3 can in principle be reduced by increasing the training length, these uncertainties are intrinsic to the methodology.

In figure 5 we show the calculation of the integral of equation (21) for different values of Q within the training range. The circuits are simulated with shot-noise since we perform $N_{\text{shots}} = 10^6$ to compute each expected value (circuit is called N_{shots} times for each estimation of the primitive G). We then repeat every measurement of the integration $N_{\text{runs}} = 100$ times. The error is computed by taking the standard deviation σ of the measurement. The shaded band in figure 5 correspond to the 1σ band.

This corresponds to an ideal real-life scenario since we are not considering hardware noise. The shot-noise error scales as $\frac{1}{\sqrt{N_{\text{runs}}}}$. This statistical noise leads in this case to an uncertainty of about 1%. Due to the computational cost we do not include in this case the training uncertainty, which would need to be included (possibly added in quadrature). We also note a systematic shift with respect to the true value of ~ 5

3.3. Normalized by construction
In this section we detail an application of our method to an actual physics problem for which the improvement with respect to classical approaches (e.g. numerical integration using MC methods) is immediate: the determination of PDFs using quantum computers [58]. We leave the actual implementation to future work, but we outline here the methodology and expected gains. Note that the same strategy can be

Figure 5. Above, integral of $xu(x,Q)$ calculated for $N_q = 20$ values of Q. These results are obtained by simulating circuits with shot-noise. In particular, for each q we perform $N_{\text{runs}} = 100$ predictions and each prediction is obtained by executing the circuit $N_{\text{shots}} = 10^6$ times. The orange line and its confidence belt are calculated using mean and one standard deviation over the N_{runs} prediction sets. Below, average relative percentage error calculated using the N_{runs} predictions. The training has been performed on approx. 100 different values of Q evenly spaced on Q^2 and took about 20 h in a 32-cores machine.

applied whenever the integral of the function is part of the fitting process. In references [59, 62] the PDFs are parametrized at $Q_0 = 1.67$ GeV such that

$$V(x) = \frac{3\hat{V}(x)}{\int_{x_a}^{x_b} dx \hat{V}(x)}, \tag{22}$$

where $V(x)$ ($\hat{V}$) corresponds to the valence (unnormalized) PDF in the so-called 'evolution basis'. and which can be written in terms of the quark and antiquark PDFs as,

$$V(x) = \sum^{u,d,s,c} q(x) - \bar{q}(x). \tag{23}$$

Note that at the parametrization energy only the lighter quarks (u, d, s and c) PDFs are independent while the heavier b and t quarks can be determined from the quarks (and gluon) PDFs.

The integral in the denominator of equation (22) is computed numerically over x between the extremes x_a and x_b and is computed from scratch for every training step. Consequently, obtaining one single value for $V(x)$ requires a costly numerical integral (more than 10^3 function calls) for each step of the training process.

With the `QiNNtegrate` approach we can instead construct a PDF which is already normalized:
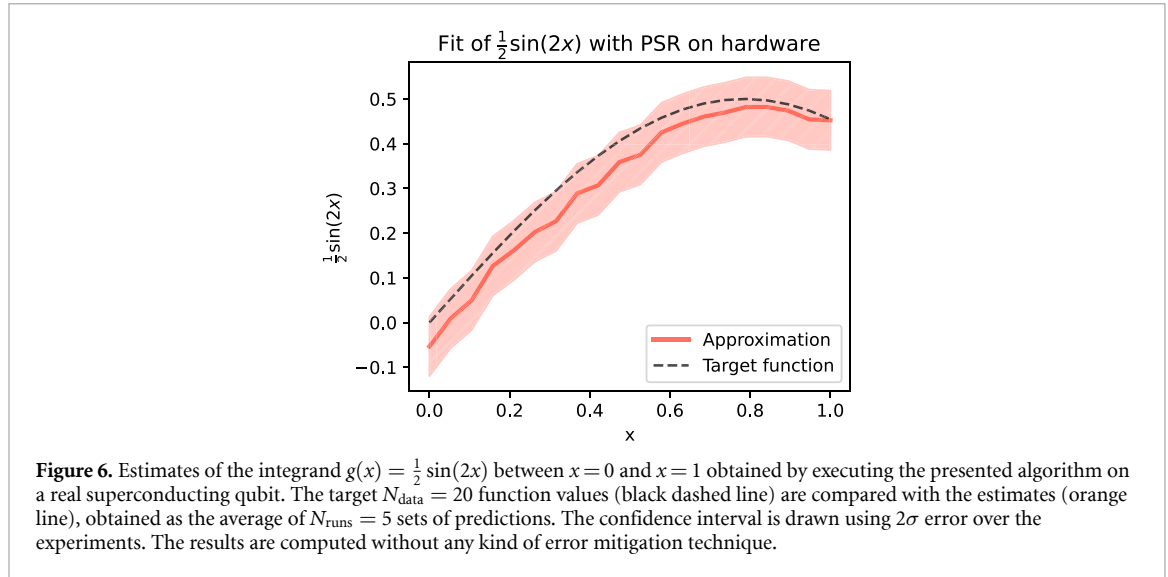
$$V(x) = \frac{\hat{V}}{I_{\hat{V}}} = \frac{3 \sum_s^{\text{shifts}} G(x_s)}{G(x_b) - G(x_a)}, \tag{24}$$

where we have exchanged the computational burden of computing the numerical integrand by a costlier evaluation of the unnormalized distribution. The shifts in the sum corresponds to all pairs of (x_+, x_-) needed to compute the derivative as per equation (9).

For the ansatz used in this work we would need 16 circuit executions per point in x involved in the fit to estimate the derivative but have removed a number of executions in the order of 10^3 from the training process. In the concrete case of reference [62] this can result in a net reduction of function calls of a factor of approximately 6. Furthermore, the fit would in this case benefit from a simpler functional form.

In addition, novel proposals for non-demolition measurements can be integrated in this algorithm to further reduce the number of shifts required to estimate the integrand function. To give an example, in [68, 69] the authors introduce an algorithm to reduce the number of expectation values required to reconstruct the gradient formula originally introduced in [20]. This is done by exploiting ancillary qubits and encoding the required shifts into a single circuit.

Together with the improvement of the qubit's quality, we believe such techniques can help in making our proposed integration algorithm even more useful in practice.

Figure 6. Estimates of the integrand $g(x) = \frac{1}{2}\sin(2x)$ between $x = 0$ and $x = 1$ obtained by executing the presented algorithm on a real superconducting qubit. The target $N_{\text{data}} = 20$ function values (black dashed line) are compared with the estimates (orange line), obtained as the average of $N_{\text{runs}} = 5$ sets of predictions. The confidence interval is drawn using 2σ error over the experiments. The results are computed without any kind of error mitigation technique.

3.4. Integrating on a real qubit

In this section we present some results obtained executing this algorithm on a real quantum hardware. In particular, we use a superconducting device composed of a single qubit hosted at the Quantum Research Center (QRC) of the Technology Innovation Institute (TII). The entire process is realized using the `Qibo` [24–30] ecosystem; the high level code is written with `Qibo` and then executed on the qubit by `Qibolab` [29]. In case we make use of `Qibosoq` [70], which is the server that integrates Qick [71] in the `Qibolab` ecosystem for executing arbitrary circuits and pulse sequences through RFsoC FPGA boards. All the single qubit characterization and calibration routines are performed using `Qibocal` [27, 30].

We tackle a simple example to reduce the number of expected values of the form of equation (4) to be evaluated. In fact, since the derivative of a circuit is used as predictor in our model, $2 * N_x$ expected values are needed to compute each estimation, where N_x is the number of times x is uploaded into the model. For example, in case of the one dimensional u quark PDF presented in section 3.2 $N_x = 2N_{\text{layers}}$. We present in figure 6 predictions of $N_{\text{data}} = 20$ values of the integrand $g(x) = \frac{1}{2}\sin(2x)$, considering N_{data} values of x uniformly distributed in $[0, 1]$. We calculate $N_{\text{runs}} = 5$ times the prediction for each x and use the mean and 2σ to define the confidence intervals around the estimates which are shown in figure 6. During the process, each expectation value is obtained executing the circuit $N_{\text{shots}} = 5000$ times. This simple example proves a simple integrand function can be fitted on a NISQ device.

As second test, we calculate the integral value of the target function:

$$I_{\text{target}} = \int_0^1 \frac{1}{2}\sin(2x)\mathrm{d}x. \tag{25}$$

$N_{\text{int}} = 10$ times obtaining as estimate: $\hat{I}_{\text{target}} = 0.326 \pm 0.011$, to be compared with the exact value $I_{\text{target}} = 0.354$. The error on the estimate is the standard deviation over the N_{int} results. We believe having such a satisfactory result even with noisy hardware can be motivated by the nature of the problem we are tackling. In fact, the target integral is here calculated following equation (12), and the difference between estimations can help in removing systematic errors that may occur when dealing with NISQ devices.

4. Conclusion

In this paper we have extended the methods proposed in references [13, 14] to quantum computers and shown how the properties of these new type of devices can introduce a practical advantage compared to classical alternatives by exploiting the properties of the parameter shift rule.

Furthermore, we have demonstrated a practical-case in which one can obtain a net benefit by utilizing this approach, by skipping entirely the need for a numerical integration during a fitting procedure. A natural extension of this work is an enhancement of the methodology proposed in reference [58] in which the PDF fit could not be normalized due to technical constraints.

We then reported interesting results obtained on a real superconducting qubit, showing how a NISQ device can already be used to fit integrand functions following our algorithm. We have also made all code available in a public python framework `QiNNtegrate` [72]. which can be used to reproduce the results of

this work and which can be extended to other custom functions. `QiNNtegrate` is based on `Qibo` and thus the generated circuits can be either simulated in a classical computer or directly executed on hardware.

Data availability statement

No new data were created or analysed in this study.

Acknowledgments

ORCID iDs

Juan M Cruz-Martinez ● https://orcid.org/0000-0002-8061-1965
Matteo Robbiati ● https://orcid.org/0009-0005-8605-234X
Stefano Carrazza ● https://orcid.org/0000-0002-0079-6753

References

[1] Metropolis N and Ulam S 1949 *J. Am. Stat. Assoc.* **44** 335
[2] Caflisch R E 1998 *Acta Numer.* **7** 1–49
[3] Zhong H and Feng X 2022 An efficient and fast sparse grid algorithm for high-dimensional numerical integration (arXiv: 2210.14313)
[4] Ghahramani Z and Rasmussen C 2002 *Advances in Neural Information Processing Systems* vol 15, ed S Becker, S Thrun and K Obermayer (MIT Press)
[5] Schmidhuber J 2015 *Neural Netw.* **61** 85
[6] Lepage G P 1978 *J. Comput. Phys.* **27** 192
[7] Lepage G P 2021 *J. Comput. Phys.* **439** 110386
[8] Cruz-Martinez J M and Carrazza S 2020 *Comput. Phys. Commun.* **254** 107376
[9] Gómez P, Toftevaag H and Meoni G 2021 *J. Open Source Softw.* **6** 3439
[10] Kleiss R and Pittau R 1994 *Comput. Phys. Commun.* **83** 141
[11] Müller T, Mcwilliams B, Rousselle F, Gross M and Novák J 2019 *ACM Trans. Graph.* **38** 1–9
[12] Bothmann E, Janen T, Knobbe M, Schmale T and Schumann S 2020 *SciPost Phys.* **8** 069
[13] Lindell D B, Martel J N P, Wetzstein G 2020 (arXiv: 2012.01714)
[14] Maitre D and Santos-Mateos R 2023 *J. High Energy Phys.* **2023** 221
[15] Schuld M, Sinayskiy I and Petruccione F 2014 *Contemp. Phys.* **56** 172
[16] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 *Nature* **549** 195
[17] Mitarai K, Negoro M, Kitagawa M and Fujii K 2018 *Phys. Rev.* A **98** 032309
[18] Chen S Y-C, Yang C-H H, Qi J, Chen P-Y, Ma X and Goan H-S 2020 Variational quantum circuits for deep reinforcement learning (arXiv: 1907.00397)
[19] Abbas A, Sutter D, Zoufal C, Lucchi A, Figalli A and Woerner S 2021 *Nat. Comput. Sci.* **1** 403
[20] Schuld M, Bergholm V, Gogolin C, Izaac J and Killoran N 2019 *Phys. Rev.* A **99** 032331
[21] Crooks G E 2019 Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition (arXiv: 1905.13311)
[22] Mari A, Bromley T R and Killoran N 2021 *Phys. Rev.* A **103** 012405
[23] Wierichs D, Izaac J, Wang C and Lin C Y-Y 2022 *Quantum* **6** 677
[24] Efthymiou S, Ramos-Calderer S, Bravo-Prieto C, Perez-Salinas A, Garc-Martin D, Garcia-Saez A, Latorre J I and Carrazza S 2021 *Quantum Sci. Technol.* **7** 015018
[25] Efthymiou S, Lazzarin M, Pasquale A and Carrazza S 2022 *Quantum* **6** 814
[26] Carrazza S, Efthymiou S, Lazzarin M and Pasquale A 2023 *J. Phys.: Conf. Ser.* **2438** 012148
[27] Pasquale A, Efthymiou S, Ramos-Calderer S, Wilkens J Roth I and Carrazza S 2023 Towards an open-source framework to perform quantum calibration and characterization (arXiv: 2303.10397)
[28] Efthymiou S *et al* 2023 qiboteam/qibo: Qibo 0.1.12 (available at: https://doi.org/10.5281/zenodo.7736837)
[29] Efthymiou S *et al 2023*, qiboteam/qibolab: Qibolab 0.0.2 (available at: https://doi.org/10.5281/zenodo.7748527)
[30] Pasquale A *et al 2023*, qiboteam/qibocal: Qibocal 0.0.1 (available at: https://doi.org/10.5281/zenodo.7662185)
[31] Preskill J 2018 *Quantum* **2** 79
[32] Delgado A *et al* 2022 Quantum computing for data analysis in high energy physics (arXiv:2004.06252)
[33] Gustafson G, Prestel S, Spannowsky M and Williams S 2022 *JHEP* **11** 035
[34] Agliardi G, Grossi M, Pellen M and Prati E 2022 *Phys. Lett.* B **832** 137228
[35] Bauer C W *et al* 2023 *PRX Quantum* **4** 027001
[36] Woźniak K A, Belis V, Puljak E, Barkoutsos P, Dissertori G, Grossi M, Pierini M, Reiter F, Tavernelli I and Vallecorsa S 2023 Quantum anomaly detection in the latent space of proton collision events at the lhc (arXiv: 2301.10780)
[37] Chawdhry H A and Pellen M 2023 (arXiv: 2303.04818)
[38] Robbiati M, Cruz-Martinez J M and Carrazza S 2023 Determining probability density functions with adiabatic quantum computing (arXiv: 2303.11346)
[39] D'Elia A *et al* 2024 *Appl. Sci,* **14** 1478
[40] Benedetti M, Lloyd E, Sack S and Fiorentini M 2019 *Quantum Sci. Technol.* **4** 043001

[41] Cerezo M, *et al* 2020 Variational quantum algorithms notecite arxiv:2012.09265Comment: Review Article. 29 pages 6 figures (arXiv: 2012.09265)
[42] Lloyd S, Schuld M, Ijaz A, Izaac J and Killoran N 2020 Quantum embeddings for machine learning (arXiv:2001.03622)
[43] Pérez-Salinas A, Cervera-Lierta A, Gil-Fuster E and Latorre J I 2020 *Quantum* **4** 226
[44] Incudini M, Martini F and Pierro A D 2022 Structure learning of quantum embeddings (arXiv: 2209.11144)
[45] Schuld M and Petruccione F 2018 *Supervised Learning With Quantum Computers* (*Quantum Science and Technology*) (Springer)
[46] Rumelhart D E, Hinton G E and Williams R J 1986 *Nature* **323** 533
[47] Kingma D P and Ba J 2017 Adam: A method for stochastic optimization (arXiv: 1412.6980)
[48] Duchi J, Hazan E and Singer Y 2011 *J. Mach. Learn. Res.* **12** 2121
[49] Ruder S 2017 An overview of gradient descent optimization algorithms (arXiv: 1609.04747)
[50] Robbiati M, Efthymiou S, Pasquale A Carrazza S 2022 A quantum analytical adam descent through parameter shift rule using qibo (arXiv: 2210.10787)
[51] Hansen N 2023 The cma evolution strategy: A tutorial (arXiv: 1604.00772)
[52] Henderson D, Jacobson S and Johnson A 2006 *The Theory and Practice of Simulated Annealing* pp 287–319
[53] Kübler J M, Arrasmith A, Cincio L and Coles P J 2020 *Quantum* **4** 263
[54] Arrasmith A, Cincio L, Somma R D and Coles P J 2020 Operator sampling for shot-frugal optimization in variational algorithms (arXiv:2004.06252)
[55] Menickelly M, Ha Y and Otten M 2023 *Quantum* **7** 949
[56] Stokes J, Izaac J, Killoran N and Carleo G 2020 *Quantum* **4** 269
[57] Pérez-Salinas A, López-Núñez D, García-Sáez A, Forn-Díaz P and Latorre J I 2021 *Phys. Rev. A* **104** 012405
[58] Pérez-Salinas A, Cruz-Martinez J, Alhajri A A and Carrazza S 2021 *Phys. Rev. D* **103** 034027
[59] Ball R D *et al* 2021 NNPDF *Eur. Phys. J. C* **81** 958
[60] Candido A, Del Debbio L, Giani T and Petrillo G 2024 (available at: https://arxiv.org/abs/2404.07573)
[61] Forte S and Watt G 2013 *Annu. Rev. Nucl. Part. Sci.* **63** 291
[62] Ball R D *et al* 2022 NNPDF collaboration *Eur. Phys. J. C* **82** 428
[63] Banchi L and Crooks G E 2021 *Quantum* **5** 386
[64] Powell M J D 1964 *Comput. J.* **7** 155
[65] Liu D C and Nocedal J 1989 *Math. Program.* **45**
[66] Wales D J and Doye J P K 1997 *J. Phys. Chem.* A **101** 5111
[67] Abbas A, King R, Huang H-Y, Huggins W J, Movassagh R, Gilboa D and McClean J R 2023 On quantum backpropagation, information reuse, and cheating measurement collapse misc (arXiv: 2305.13362)
[68] Solinas P, Caletti S and Minuto G 2023 *Eur. Phys. J. D* **77** 76
[69] Minuto G, Caletti S and Solinas P 2024 A novel approach to reduce derivative costs in variational quantum algorithms (arXiv: 2404.02245)
[70] Carobene R, Candido A, Serrano J and Carrazza S 2023 Edoardo-Pedicillo, qiboteam/qibosoq: qibosoq 0.0.3 (https://doi.org/10.5281/zenodo.8126172)
[71] Stefanazzi L *et al 2022* The qick quantum instrumentation control kit: readout and control for qubits and detectors (arXiv: 2110.00557)
[72] Martinez J M C, Robbiati M and Carrazza S 2023 QiNNtegrate (available at: https://github.com/qiboteam/QiNNtegrate)