Quantum Quantile Mechanics: Solving Stochastic Differential Equations for Generating Time-Series

Annie E. Paine,[1,2] Vincent E. Elfving,[1] and Oleksandr Kyriienko[1,2]

[1]*Qu & Co B.V., PO Box 75872, 1070 AW, Amsterdam, The Netherlands*
[2]*Department of Physics and Astronomy, University of Exeter, Stocker Road, Exeter EX4 4QL, United Kingdom*

(Dated: August 9, 2021)

We propose a quantum algorithm for sampling from a solution of stochastic differential equations (SDEs). Using differentiable quantum circuits (DQCs) with a feature map encoding of latent variables, we represent the quantile function for an underlying probability distribution and extract samples as DQC expectation values. Using quantile mechanics we propagate the system in time, thereby allowing for time-series generation. We test the method by simulating the Ornstein-Uhlenbeck process and sampling at times different from the initial point, as required in financial analysis and dataset augmentation. Additionally, we analyse continuous quantum generative adversarial networks (qGANs), and show that they represent quantile functions with a modified (reordered) shape that impedes their efficient time-propagation. Our results shed light on the connection between quantum quantile mechanics (QQM) and qGANs for SDE-based distributions, and point the importance of differential constraints for model training, analogously with the recent success of physics informed neural networks.

INTRODUCTION

Stochastic differential equations (SDEs) describe a broad range of phenomena. They emerge when dealing with Brownian motion and quantum noise [1]. In physical sciences, SDEs are used for describing quantum dynamics [2], thermal effects [3], molecular dynamics [4], and they lie at the core of stochastic fluid dynamics [5, 6]. In biology SDEs help in the studies of population dynamics [7] and epidemiology [8, 9]. They can also help to detect anomalies [10]. SDEs are widely used in financial calculus [11], a fundamental component of all mechanisms of pricing financial derivatives and description of market dynamics. Potential applications of SDEs in finance lie in predicting stock prices, currency exchange rates and more [12].

A general system of stochastic differential equations can be written as [11]

$$d\boldsymbol{X}_t = f(\boldsymbol{X}_t, t)dt + g(\boldsymbol{X}_t, t)d\boldsymbol{W}_t, \tag{1}$$

where $\boldsymbol{X}_t$ is a vector of stochastic variables parameterized by time t (or other parameters). Deterministic functions f and g correspond to the drift and diffusion processes, respectively. $\boldsymbol{W}_t$ corresponds to the stochastic Wiener process. The stochastic component makes SDEs distinct from other types of partial differential equations, adding a non-differentiable contribution. This also makes SDEs generally difficult to treat. Several questions arise: how do we solve Eq. (1), and what kind of information do we want to get by solving SDEs? These are not trivial questions to answer, because one might be interested in different aspects of SDE-modelled processes.

First, the system of SDEs can be rewritten in the form of a partial differential equation for the underlying probability density function (PDF) $p(\boldsymbol{x}, t)$ of deterministic continuous variables, and then solved for the specified boundary conditions. The resulting equation is known as a Fokker-Planck (FP) or Kolmogorov equation [13]. It can be exploited for calculating observables (averages) from $p(\boldsymbol{x}, t)$; in many cases these are the mean and the variance for the stochastic variables, $\mathrm{E}[\boldsymbol{X}_t] = \int \boldsymbol{x} p(\boldsymbol{x}, t)d\boldsymbol{x}$ and $\mathrm{Var}[\boldsymbol{X}_t] = \mathrm{E}[\boldsymbol{X}_t^2] - \mathrm{E}[\boldsymbol{X}_t]^2$, respectively. Here, a challenge arises when a well-defined boundary condition is missing, and instead some additional data is available. The task then falls into the category of data-driven machine learning problems, which has attracted attention recently [14, 15]. Also, a (potentially implicit) solution of the FP equation does not offer strategies to generate samples directly. Namely, drawing $\boldsymbol{x} \sim p(\boldsymbol{x}, t)$ from a complicated multidimensional distribution, at different t, represents another computationally expensive problem to solve.

Second, Eq. (1) can be integrated using the Euler-Maruyama method [16], where the deterministic part of the differential equation is solved with stepwise propagation, and the Wiener process is modelled with a random number generator. This corresponds to the ensemble generation process (broadly speaking, *generative modelling*), which is computationally challenging for complex models. For multidimensional systems this is hindered by the curse of dimensionality. The task of solving random partial differential equations was recently addressed with deep learning using physics-informed neural networks [17–19]. Unsupervised generative modelling often requires using adversarial training [20, 21] and generative adversarial network (GAN) architecture [22]. Moreover, in cases where the information about system parameters is limited, or exemplary datasets are generated from measurement and not known initial conditions, we arrive at the task of equation discovery [23–27].

Quantum computers operate in a multidimensional state space and are intrinsically probabilistic. They offer computational advantage for sampling tasks [28–31]. This was demonstrated experimentally with superconducting [32] and optical [33] quantum devices. Thus, quantum computing may be a prime candidate for disrupting the field of generative modelling. Yet, the challenge of performing the quantum generative modelling for practical tasks remains open. Ideally, discrete distributions can be loaded into a quantum register using an amplitude encoding $\sum_{k=1}^{K} u_k|k\rangle$, where amplitudes u_k contain information about the probability distribution and $|k\rangle$ are binary states for $\log(K)$-qubit register. This allows for a quadratic speed-up when using the amplitude amplification protocol [34], and has numerous financial use-cases [35].

However, representing arbitrary distributions in an amplitude-encoded form may be difficult (no scalable scheme exists yet), and the processing requires a large-scale fault-tolerant quantum computer [36]. A strategy that is viable for near- and mid-term QC relies on variational quantum algorithms based on parametrized quantum circuits [37, 38]. These can be seen as quantum neural networks (QNNs) [39–43]. To date, several variational protocols for solving differential equations have been proposed for ordinary and partial differential equations [44–47], targeting near-term devices. For stochastic differential equations several algorithms have been explored that are based on wavefunction-based probability encoding [48–51] and rely on complicated circuits. A strategy for near- and mid-term devices, SDE-based generative modelling can be addressed by data-based learning of probability distributions.

Motivated by classical GAN successes [22, 52], various protocols for the adversarial training of generative quantum models were proposed, and coined as *qGANs* [53, 54]. Here, several sample generation strategies can be employed. One possible strategy relies on the variational wavefunction preparation as a distribution proxy, with the sample readout following the Born rule. This corresponds to a quantum circuit Born machine (QCBM) generator [55]. In this case qGAN was applied to sampling from discrete distributions [56–62], preparing arbitrary quantum states [63, 64], and being demonstrated experimentally with superconducting circuits for image generation [65]. In this setting the sampling procedure is efficient, but its power depends on the register width and the generator is difficult to train at increasing scale. The latter comes from demanding requirements on the training set [66] and barren plateaus for global cost functions [67]. We also note that QCBM can be trained in other ways, including maximum mean discrepancy and various statistical divergences [55, 68–70], with the goal of preparing distributions arising in financial applications. The second possible strategy relies on a QNN-based generator representing a *continuous qGAN* [66], recently demonstrated experimentally [71], where a feature map embedding [72] is used for continuous latent variable distribution. In this case the sampling rate is reduced due to the expectation value measurement, but the model becomes trainable, and its power is ultimately limited by the properties of feature maps and the adversarial training schedule (i.e. minimax game instead of loss minimization). The operation of qGANs was recently surveyed in Ref. [73]. Finally, other generator architectures are represented by quantum Boltzmann machines [74–76] where sampling is based on thermal states of quantum Ising Hamiltonian, or recurrent neural networks for NMR quantum computers [77].

We note that the prior art in both classical and quantum generative modelling is either concentrated on representing probability density functions in model-driven approaches, or training the generator in a black-box fashion [e.g. (q)GANs] where no physical constraints are added. In this work, we suggest to shift the focus to the very tool that enables sampling — the *quantile function* (QF) (not to be mistaken with the quantile of a distribution). We propose to solve SDEs by rewriting them as differential equations for the quantile function and using their neural representation (classical or quantum neural networks). In the following, we use feature maps and differentiable quantum circuits (DQCs) to represent directly the quantile function of the probability distribution of the underlying SDE, and propagate them in time by solving the differential equations of quantile mechanics. This allows us to prepare a QNN-based generator that is trained from available data and yet is model-informed. While the expectation-based readout associated to the QNN structure requires multiple shots, the proposed approach has improved trainability compared to QCBM architecture and can work with sparse training data. Specifically, we benchmark the developed quantum quantile mechanics (QQM) approach using the Ornstein-Uhlenbeck model (a prototypical stochastic process being the base of many financial calculations). We show how to train QF from data and/or the known model at the initial point of time, and find a time-propagated QF that enables high-quality sampling. We then proceed to show that adversarial schemes (continuous GAN or qGAN) in fact train generators as a *reordered quantile function*. Analyzing the similarities and differences between the two methods, we find that quantile functions in their original meaning are suitable for time propagation, while reordered QFs have apparent difficulties for the task (see Discussion). Our work uncovers the possibilities for time series generation and *data-augmentation* enhanced by quantum resources.

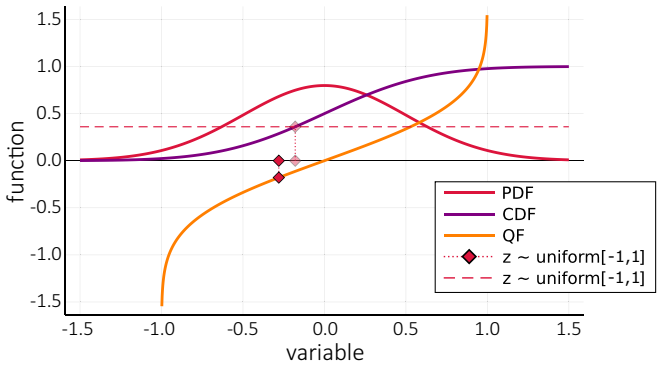

FIG. 1: **Illustrative example of sampling.** We plot a normal probability density function as a red curve (PDF), being the Gaussian function with $(\mu, \sigma) = (0, 1/2)$. The domain is chosen as $\mathcal{X} = [-1.5, 1.5]$. The corresponding cumulative distribution function is presented by the purple solid curve (CDF). The quantile function for the corresponding CDF is shown in orange (QF). The red diamonds correspond to a randomly drawn latent variable z, and associated probability for the sample value, connected by dotted lines.

RESULTS

We start by describing the background for generative modelling from SDEs, and proceed to introduce the proposed QQM method. In the second part of the Results section we present numerical experiments for training DQCs to represent time-dependent quantile functions, as well as qGAN training.

Classical sampling and quantile mechanics

Let us recall how a sample from a distribution can be obtained using a basic inversion sampling. In the following we consider

a single stochastic process X_t (also written as X for brevity), and the generalization to multiple processes/dimensions is straightforward. First, we take a PDF $p(x)$ of a continuous variable $x \in \mathcal{X}$ being normalized over domain $\mathcal{X}$. Next, we find a cumulative distribution function (CDF) for the stochastic variable X defined as an integral $F_X(x) = \int_{-\infty}^{x} p(x')dx'$. This maps x to the probability value lying in $[0, 1]$ range on the ordinate axis (see Fig. 1 for an illustration). Being a non-decreasing function, $F_X(x)$ has to be inverted to provide a sample. Generating a random number $z \in (-1, 1)$ from the uniform distribution, we can solve the equation $z = F_X(x)$ and find the corresponding sample. This requires finding the inverse of CDF, $F_X^{-1}(x)$. In most cases this leads to a transcendental problem without a closed-form solution. This poses computational challenges and requires graphical solution methods [78]. Finally, each random number $z \sim \texttt{uniform}(-1, 1)$ gives a random sample X from PDF of interest as $X = F_X^{-1}(z)$ (note that the range of z can be easily rescaled). The inverted CDF function is known as a *quantile function* of the continuous distribution, $F_X^{-1}(z) \equiv Q(z)$.

While generally quantile functions are difficult to get from PDFs, they can be obtained by solving nonlinear partial differential equations derived from SDEs of interest. This approach is called the *quantile mechanics*. The quantile function $Q(z, t)$ can be obtained for any general SDE in the form (1) and its evolution is given by quantilized FP as (see full derivation in Ref. [79])

$$\frac{\partial Q(z,t)}{\partial t} = f(Q,t) - \frac{1}{2}\frac{\partial g^2(Q,t)}{\partial Q} + \frac{g^2(Q,t)}{2}\left(\frac{\partial Q}{\partial z}\right)^{-2}\frac{\partial^2 Q}{\partial z^2}, \tag{2}$$

where $f(Q, t)$ and $g(Q, t)$ are the drift and diffusion terms familiar from Eq. (1). Eq. (2) is solved as a function of latent variable z and time t. Once $Q(z, t)$ is known, evaluating it at random uniform z's as t progresses we can get full time series (trajectories) obeying the stochastic differential equation (1). In Supplemental Information we also consider the case of solving reverse-time SDEs.

In general, quantile mechanics equations are not easy to solve. Power series solution as function approximation are known [79–82], as well as some simple examples [83]. However, the difficulty arises when multidimensional problems are considered and generative modelling suffers from the curse of dimensionality. To harness the full power of quantile mechanics, we thus propose to use neural representation of QFs. To our knowledge, this is the first application of machine learning methods to use quantile-based sampling, and we envisage that both classical and quantum ML can be used for the universal function approximation [72, 84]. Here, the use of quantum neural networks offers a potential to reproducing complex functions in the high-dimensional space, including systems where strong correlations are important. In the following, we combine quantum computing and quantile mechanics to develop the *quantum quantile mechanics* approach. The sketch of the QQM workflow is shown in Fig. 2. We represent QFs as quantum neural networks in the DQC form, thus exploiting the large expressivity of quantum-based learning. We show how differential equations for quantile functions can be used

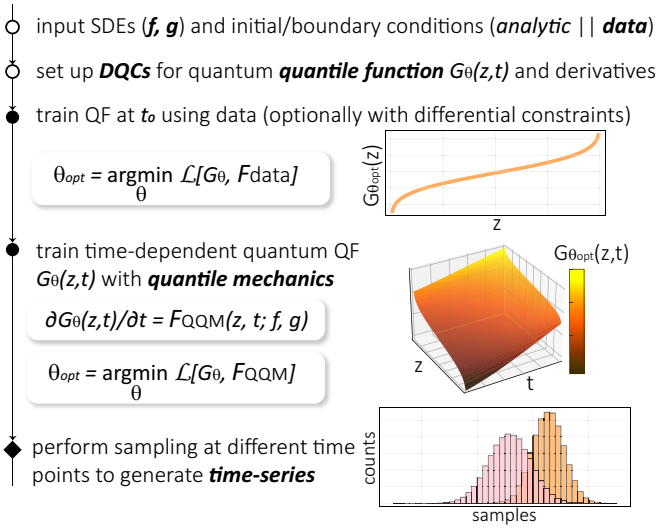

FIG. 2: **QQM workflow.** Given the system of stochastic differential equations and initial data, differentiable quantum circuits are trained to represent the corresponding quantile function $G_\theta(z, t)$ as a function of a random latent variable (z) and propagate it in time (t) with quantum mechanics equations. The hybrid quantum-classical loop is used for optimizing variational parameters θ through loss function $\mathcal{L}$ minimization based on data and differential equations for the initial and propagated QF. Evaluating $G_{\theta_{\text{opt}}}(z \sim \texttt{uniform}(-1, 1), t)$ at optimal angles, random values of the latent variable, and different time points t, we generate time series from SDE.

for training differentiable quantum circuits. Second, we introduce the quantum quantile learning protocol for inferring QF from data and use QQM to propagate the system in time. This provides a robust protocol for time series generation and sampling. Finally, we show that generative adversarial networks act as quantile functions for randomized association of the latent variable values and samples.

Quantum quantile mechanics

To represent a trainable (neural) quantile function we construct a parametrized quantum circuit using quantum embedding through feature maps $\hat{\mathcal{U}}_\phi(x)$ (with ϕ labelling a mapping function) [37, 39, 85], followed by variationally-adjustable circuit (ansatz) $\hat{\mathcal{U}}_{\boldsymbol{\theta}}$ parametrized by angles $\boldsymbol{\theta}$. The latter is routinely used in variational quantum algorithms [38], and for ML problems has the hardware efficient ansatz (HEA) structure [86]. The power of quantum feature maps comes from mapping $x \in \mathcal{X}$ from the data space to the quantum state $|\psi(x)\rangle = \hat{\mathcal{U}}_\phi(x)|\emptyset\rangle$ living in the Hilbert space. Importantly, the automatic differentiation of quantum feature maps allows derivatives to be represented as DQCs [45]. The readout is set as a sum of weighted expectation values [72]. Following this strategy, we assign a generator circuit $G(z, t)$ to represent a function parametrized by t (labels time as before), and the

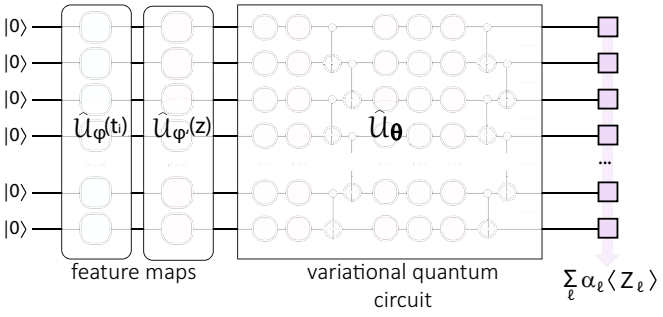

FIG. 3: **DQC layout.** State preparation circuit for representing a time-dependent quantile function. Acting on the initial state, two feature maps, with $\hat{R}_y$ using for time-dependence embedding and $\hat{R}_x$ layer for the latent variable embedding. We use HEA for the variational search, and total Z magnetization as a cost function.

embedded latent variable z. The generator reads

$$G(z,t) = \langle \emptyset | \hat{\mathcal{U}}_\phi(t)^\dagger \hat{\mathcal{U}}_{\phi'}(z)^\dagger \hat{\mathcal{U}}_{\boldsymbol{\theta}}^\dagger \Big(\sum_{\ell=1}^{L} \alpha_\ell \hat{C}_\ell \Big) \hat{\mathcal{U}}_{\boldsymbol{\theta}} \hat{\mathcal{U}}_{\phi'}(z) \hat{\mathcal{U}}_\phi(t) | \emptyset \rangle, \tag{3}$$

where $\hat{\mathcal{U}}_\phi(z)$ and $\hat{\mathcal{U}}_{\phi'}(t)$ are quantum feature maps (possibly different), $\{\hat{C}_\ell\}_{\ell=1}^L$ represent L distinct Hermitian cost function operators, and $\{\alpha_\ell\}_{\ell=1}^L$ and $\boldsymbol{\theta}$ are real coefficients that may be adjusted variationally. We denote the initial state as $|\emptyset\rangle$, typically chosen as a product state $|0\rangle^{\otimes N}$. The circuit structure is shown in Fig. 3. We can also work with multiple latent variables and thus multidimensional distributions.

Our next step is developing the training procedure for G [or specifically for the generator's operator $\hat{\mathcal{U}}_G(z) = \hat{\mathcal{U}}_{\boldsymbol{\theta}} \hat{\mathcal{U}}_{\phi'}(z)$], such that it represents QF for an underlying data distribution. Namely, we require that the circuit maps the latent variable $z \in [-1, 1]$ to a sample $G(z) = Q(z)$. Choosing different sets of cost functions, with the same quantum circuits, we can produce samples from multidimensional PDF.

The training requires a dataset $\{X_{\text{data}}\}$ generated from a probability distribution for the system we want to study (or measured experimentally), which serves as an initial/boundary condition. Additionally, we know the underlying processes that describe the system and which serve as differential constraints. The problem is specified by SDEs $d\boldsymbol{X}_t = f_{\boldsymbol{\xi}}(\boldsymbol{X}_t, t)dt + g_{\boldsymbol{\xi}}(\boldsymbol{X}_t, t)d\boldsymbol{W}_t$, where we explicitly state that the drift and diffusion functions $f_{\boldsymbol{\xi}}(\boldsymbol{X}_t, t)$ and $g_{\boldsymbol{\xi}}(\boldsymbol{X}_t, t)$ are parametrized by the vector $\boldsymbol{\xi}$ (time-independent). We consider that the SDE parameters for generating data similar to $\{X_{\text{data}}\}$ are known in the first approximation $\boldsymbol{\xi}^{(0)}$. This can be adjusted during the training to have the best convergence, as used in the equation discovery approach [23, 25]. The loss is constructed as a sum of data-based and SDE-based contributions, $\mathcal{L} = \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{SDE}}$. The first part $\mathcal{L}_{\text{data}}$ is designed such that the data points in $\{X_{\text{data}}\}$ are represented by the trained QF. For this, the data is binned appropriately and collected in ascending order, as expected for any quantile function. Then, we use quantum circuit learning (QCL) as a quantum nonlinear regression method [85] to learn the quantile function. We note that such approach represents a data-frugal strategy, where the need for training on *all* data points is alleviated. The second loss term $\mathcal{L}_{\text{SDE}}$ is designed such that the learnt quantile function obeys probability models associated to SDEs. Specifically, the generator $G(z,t)$ needs to satisfy the quantilized Fokker-Planck equation (2). The differential loss is introduced using the DQC approach [45], and reads

$$\mathcal{L}_{\text{SDE}} = \frac{1}{M} \sum_{z,t \in \mathcal{Z},\mathcal{T}} \mathfrak{D}\left[\frac{\partial G_{\boldsymbol{\theta}}}{\partial t}, F(z,t,f,g, \frac{\partial G_{\boldsymbol{\theta}}}{\partial z}, \frac{\partial^2 G_{\boldsymbol{\theta}}}{\partial z^2}) \right], \tag{4}$$

where $\mathfrak{D}[a,b]$ denotes the distance measure for two scalars, and the loss is estimated over the grid of points in sets $(\mathcal{Z}, \mathcal{T})$. Here $M = \text{car}(\mathcal{Z})\text{car}(\mathcal{T})$ is the total number of points. We also introduce the function $F(z,t,f,g,\partial_z G_{\boldsymbol{\theta}}, \partial^2_{zz} G_{\boldsymbol{\theta}})$ denoting the RHS for Eq. (2), or any other differential constraint. Other important ingredients of the QQM method include the calculation of second-order derivatives for the feature map encoded functions (as required is $\mathcal{L}_{\text{SDE}}$), and the proposed treatment of initial/boundary conditions for multivariate function. We describe these technical details in the Methods section. Once the training is set up, the loss is minimized using a hybrid quantum-classical loop where optimal variational parameters $\boldsymbol{\theta}_{\text{opt}}$ (and $\boldsymbol{\alpha}_{\text{opt}}$) are searched using non-convex optimization methods.

QQM-based generative modelling

In the next subsections we present numerical simulations of generative modelling. In the first part we apply the developed quantum quantile mechanics approach for solving a specific SDE, and demonstrate a data-enabled operation. In the second part we cover the so-called quantum generative adversarial network (qGAN) that was previously used for continuous distributions, and show numerical results for solving the same problem. The two approaches are then compared in the next section (Discussion).

Ornstein-Uhlenbeck for financial forecasting and trading. To validate the QQM approach and perform time series forecasting, we pick a prototypical test problem. As an example we choose the Ornstein-Uhlenbeck (OU) process. In financial analysis its generalization is known as the Vasicek model [87]. It describes the evolution of interest rates and bond prices [88]. This stochastic investment model is the time-independent drift version of the Hull–White model [89] widely used for derivatives pricing. We also note that OU describes dynamics of currency exchange rates, and is commonly used in Forex pair trading — a primary example for quantum generative modelling explored to date [69, 70]. Thus, by benchmarking the generative power of QQM for OU we can compare it to other strategies (valid at fixed time point).

The Ornstein-Uhlenbeck process is described by an SDE with an instantaneous diffusion term and linear drift. For a single variable process X_t the OU SDE reads

$$X_t = \nu(\mu - X_t)dt + \sigma dW_t, \tag{5}$$

where the vector of underlying parameters $\boldsymbol{\xi} = (\nu, \mu, \sigma)$ are the speed of reversion ν, the long-term mean level μ, and the de-

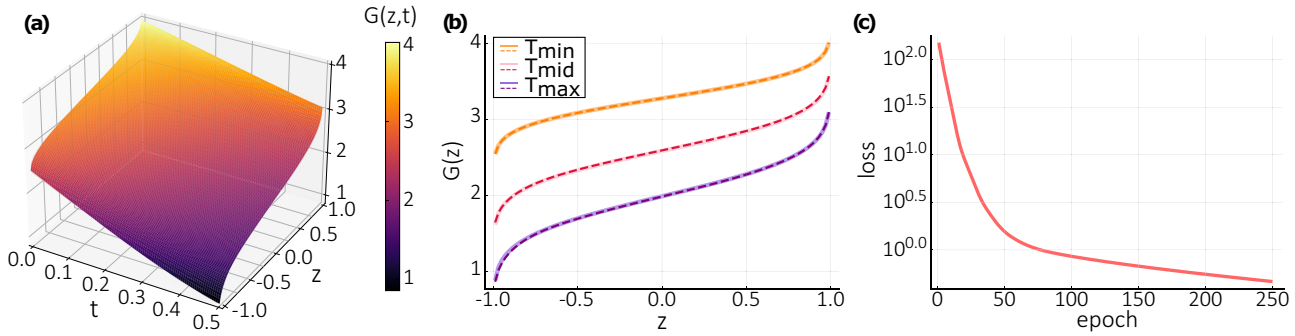

FIG. 4: **Training time evolution of Ornstein-Uhlenbeck process.** The results are shown for runs with analytic initial condition and parameters chosen as $\nu = 1, \sigma = 0.7, x_0 = 4$. **(a)** Surface plot for the trained DQC-based quantile function $G(z,t)$ that changes in time. **(b)** Slices of the quantum quantile function $G(z,t)$ shown at discrete time points $t = 0$ (labelled as T_{min} hereafter), $t = 0.25$ (T_{mid}), and $t = 0.5$ (T_{max}). **(c)** Loss as a function of epoch number showing the training progression and final loss for circuits shown in **(a, b)**.

gree of volatility σ. The corresponding Fokker-Planck equation for the probability density function $p(x,t)$ reads

$$\frac{\partial p(x,t)}{\partial t} = \nu \frac{\partial}{\partial x}(xp) + \frac{\sigma^2}{2}\frac{\partial^2 p}{\partial x^2}. \tag{6}$$

When rewritten in the quantilized form, it becomes a PDE for the quantile mechanics,

$$\frac{\partial Q(z,t)}{\partial t} = \nu[\mu - Q(z,t)] + \frac{\sigma^2}{2}\left(\frac{\partial Q}{\partial z}\right)^{-2}\frac{\partial^2 Q}{\partial z^2}, \tag{7}$$

which follows directly from the generic Eq. (2). In the following we take the speed of reversion to be positive, $\nu > 0$ and adjust the long-term mean level to zero, $\mu = 0$.

Having established the basics, we train the differentiable quantum circuit to match the OU QF. First, for the starting point of time, we train the circuit to represent a quantile function based on available data (see the workflow chart in Fig. 2 and the discussion below). Next, having access to the quantum QF at the starting point, we evolve it in time solving the equation

$$\frac{\partial G(z,t)}{\partial t} = -\nu G(z,t) + \frac{\sigma^2}{2}\left(\frac{\partial G}{\partial z}\right)^{-2}\frac{\partial^2 G}{\partial z^2}, \tag{8}$$

as required by QM [Eq. (7)]. This is the second training stage in the workflow chart shown in Fig. 2.

To check the results, we use the analytically derived PDF valid for the Dirac delta initial distribution $p(x,t_0) = \delta(x - x_0)$ peaked at x_0 that evolves as

$$p(x,t) = \sqrt{\frac{\nu}{\pi\sigma^2(1-\exp[-2\nu(t-t_0)])}}\times \\ \times \exp\left[-\frac{\nu(x - x_0\exp[-\nu(t-t_0)])^2}{\sigma^2(1-\exp[-2\nu(t-t_0)])}\right], \tag{9}$$

and we can write the OU QF evolution as

$$Q(z,t) = x_0\exp[-\nu(t-t_0)] + \\ + \sqrt{\frac{\sigma^2}{\nu}\left(1-\exp[-2\nu(t-t_0)]\right)}\,\text{inverf}(z), \tag{10}$$

where inverf(x) denotes the inverse error function [90]. This provides us a convenient benchmark of a simple case application and allows assessing the solution quality. Additionally we use Euler-Maruyama integration to compare results with the numerical sampling procedure with fixed number of shots.

Ornstein-Uhlenbeck with analytic initial condition. To

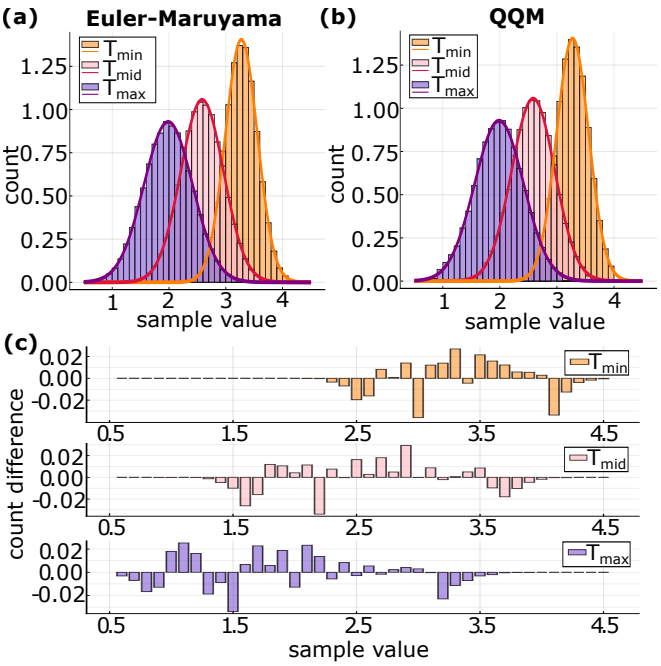

FIG. 5: **Comparison of histograms from the numerical SDE integration and QQM training.** **(a)** Distribution of samples from the Euler-Maruyama SDE solver (binned counts divided by the total number of samples N_s), shown against the analytic PDF at three time points T_{min} ($t = 0$), T_{mid} ($t = 0.25$), and T_{max} ($t = 0.5$). $N_s = 100,000$ samples are taken, and parameters are the same ($\nu = 1$, $\sigma = 0.7$, $x_0 = 4$). **(b)**. Distribution of samples generated from the DQC-based quantile function, for the training described in Fig. 4. **(c)** Bar height difference between **(a)** and **(b)** shown for T_{min} (top), T_{mid} (middle), and T_{max} (bottom).

highlight the generative power of the QQM approach we start by simulating the OU evolution $G(z,t)$ starting from the known initial condition. This is set as $G(z,0) = Q(z,0)$ being the analytic solution [Eq. (10)] or can be supplied as a list of known samples associated to latent variable values. To observe a significant change in the statistics and challenge the training, we choose the dimensionless SDE parameters as $\nu = 1$, $\sigma = 0.7$, $x_0 = 4$, and $t_0 = -0.2$ such that we evolve a narrow normal distribution with strongly shifted mean into a broad normal distribution at $\mu = 0$. We use DQCs with $N = 6$ qubits and a single cost operator being the total Z magnetization, $\hat{C} = \sum_{j=1}^{N} \hat{Z}_j$. For simplicity we train the circuit using a uniformly discretized grid with $\mathcal{Z}$ containing 21 points from -1 to 1, and $\mathcal{T}$ containing 20 values from 0.0 to 0.5. To encode the function we use the product-type feature maps [45, 85] chosen as $\hat{\mathcal{U}}_\phi(t) = \bigotimes_{j=1}^{N} \exp[-i \arcsin(t)\hat{Y}_j/2]$ and $\hat{\mathcal{U}}_{\phi'}(z) = \bigotimes_{j=1}^{N} \exp[-i \arcsin(z)\hat{X}_j/2]$. The variational circuit corresponds to HEA with the depth of six layers of generic single-qubit rotations plus nearest-neighbor CNOTs. We exploit the floating boundary handling, and choose a mean squared error (MSE) as the distance measure, $\mathfrak{D}(a,b) = (a-b)^2$. The system is optimized for a fixed number of epochs using the `Adam` optimizer for gradient-based training of variational parameters $\boldsymbol{\theta}$. We implement this with a full quantum state simulator in a noiseless setting. This is realized in `Yao.jl` [91] — a `Julia` package that offers state-of-the-art performance.

We present the results of DQC training in Fig. 4. In Fig. 4(a) we show the trained quantum QF as a function of time t and the latent variable z. Choosing three characteristic points of time $t = \{0.0, 0.25., 0.50\}$ that we label as $\{T_{\min}, T_{\mathrm{mid}}, T_{\max}\}$, we plot the corresponding quantile functions at these times [Fig. 4]. The dashed curves from the DQC training closely follow ideal QFs shown by solid curves. Additionally, in Fig. 4(c) we show the training loss as a function of epoch number, noting a rapid convergence as the circuit is expressive enough to represent changes of initial QF at increasing time, and thus providing us with evolved $G(z,t)$.

Next, we perform sampling and compare the histograms coming from the Euler-Maruyama integration of OU SDE [16, 92] and the QQM training presented above. The results are shown in Fig. 5 for the same parameters as Fig. 4. In Fig. 5(a) we show the three time slices of Euler-Maruyama trajectories, built with $N_s = 100,000$ samples to see distributions in full. The counts are binned and normalized by N_s, and naturally show excellent correspondence with analytical results. The sampling from trained quantile is performed by drawing random $z \sim$ `uniform`$(-1,1)$ for the same number of samples. In Fig. 5(b) we observe that QQM matches well the expected distributions. Importantly, the training correctly reproduces the widening of the distribution and the mean reversion, avoiding the mode collapse that hampers adversarial training [52, 73]. To further corroborate our findings, we plot the difference between two histograms (Euler-Maruyama and QQM) in Fig. 5(c), and observe that the count difference remains low at different time points.

Ornstein-Uhlenbeck with data-inferred initial condition. Next, we demonstrate the power of quantile function training

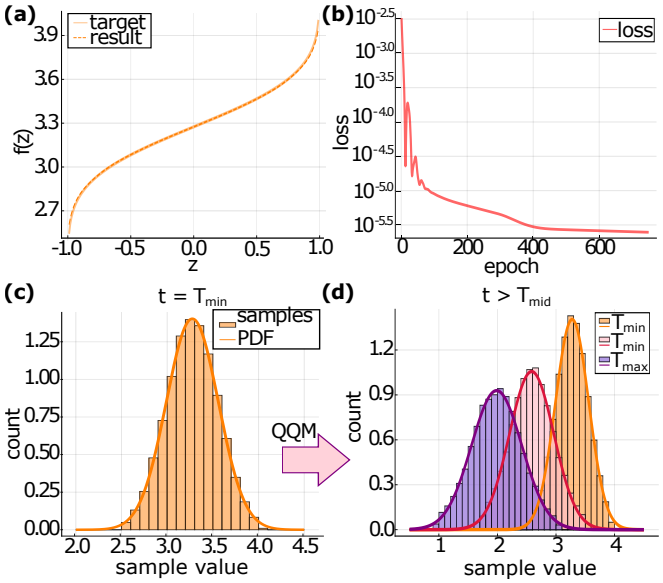

FIG. 6: **Quantile function trained on initial data.** **(a)** Trained QF for the Ornstein-Uhlenbeck process at $t = 0$ (dashed curve labeled as `result`), plotted together with the known true quantile (solid line labeled as `target`, results overlay). The parameters are the same as for Fig. 4. **(b)** Training loss at different epochs, with the final epoch producing the QF in **(a)**. **(c)** Normalised histogram of samples from the data-trained QF, plotted against the analytic distribution (PDF). $N_s = 100,000$ random samples are drawn and bin counts are normalized by N_s as before. **(d)** Histograms for the data-trained QF evolved with quantum quantile mechanics, shown at three points of time.

from the available data (observations, measurements) corresponding to the Ornstein-Uhlenbeck process. Note that compared to the propagation of a known solution that is simplified by the boundary handling procedure, for this task we learn both the surface $G(z,t)$ and the initial quantile function $G(z, T_{\min})$. To learn the initial QF (same parameters as for Figs. 4 and 5) we use QCL trained on the observations. The samples in the initial dataset are collected into bins and sorted in the ascending order as required by QF properties. From the original $N_s = 100,000$ that are ordered we obtain an interpolated curve. We get target values for QCL training choosing $N_{\mathrm{points}} = 43$ points in $\mathcal{Z}$ between -1 and 1. We note that the training set is significantly reduced, and such data-frugal training holds as long as the QF structure is captured (monotonic increase). The training points are in the Chebyshev grid arrangement as $\cos[(2n-1)\pi/(2N_{\mathrm{points}})]$ $(n = 1, 2, ..., N_{\mathrm{points}})$, this puts slight emphasis on training the distribution tails around $|z| \approx 1$. To make the feature map expressive enough that it captures full z-dependence for the trained initial QF, we use a tower-type product feature maps defined as $\hat{\mathcal{U}}_{\phi'}(z) = \bigotimes_{j=1}^{N} \exp[-i \arcsin(z) j\hat{Z}_j/2]$, where rotation angles depend on the qubit number j. For the training we again use a six-qubit register, and follow the same variational strategy as in the previous subsection. We observe that a high-quality solution with a loss of $\sim 10^{-6}$ for $G(z,0)$ can be obtained at the number of epochs increased to few thou-

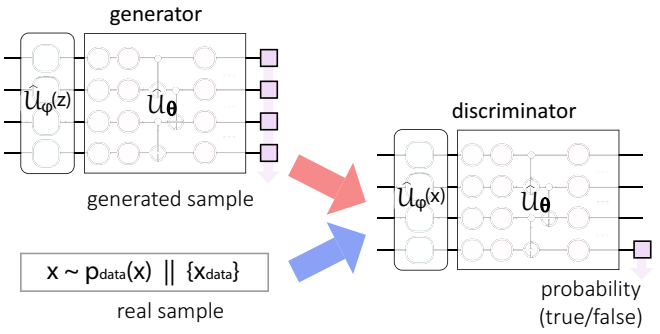

FIG. 7: **Quantum GAN workflow.** The quantum circuits are used both for generative modelling at $t = T_{\min}$ (generator) and discrimination between real and fake samples (discriminator). The generator circuit $G_{\mathrm{Q}}(z)$ is composed of the product feature map and HEA variational circuit. The discriminator $D_{\mathrm{Q}}(x)$ is trained to distinguish samples from the initial data distribution.

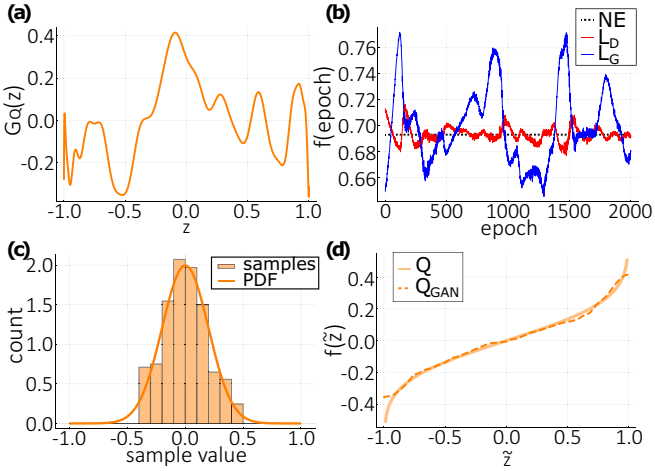

FIG. 8: **qGAN training and fixed-time sampling.** **(a)** Generator function is shown for the optimal training angles. **(b)** Generator (L_G, red) and discriminator (L_D, blue) loss terms at different epochs. The Nash equilibrium at $-\ln(1/2)$ is shown by black dotted line (NE). **(c)** Normalized histogram for qGAN sampling ($N_{\mathrm{s}} = 100,000$), as compared to the target normal distribution ($\mu = 0$, $\sigma = 0.2$). **(d)** Ordered quantile $Q_{\mathrm{GAN}}(\tilde{z})$ from the resulting qGAN generator shown in **(a)** (dashed curve), as compared to the true QF of the target distribution (solid curve).

sands, and we find that pre-training with product states allows reducing this number to hundreds with identical quality.

The results are shown in Fig. 6, with the QF trained from data shown in Fig. 6(a) by the dashed curve that overlays the target QF. The circuit converges to 10^{-6} loss level [Fig. 6(b)], being close to the model expressivity limit of the feature map itself. Performing sample generation at the initial time, in Fig. 6 we observe good correspondence with the expected PDF (sampling procedure is the same as in Fig. 5). At the same time we note that small deviations in trained QF (and its derivative) lead to significant deviations for statistics, stressing the importance of expressive circuits and stable training. Using the trained QF as the initial condition, we evolve the system as before (training details are presented in Supplemental Information). We perform generative modelling at later points of time T_{mid} and $T_{\max}$. The histograms in Fig. 6 confirm the high quality of sampling, and show that the approach is suitable for time series generation.

qGAN-based generative modelling

Generative adversarial networks represents one of the most successful strategies for generative modelling [52]. It is used in various areas, ranging from the fashion industry to finance, where GANs are used to enrich financial datasets. The latter is specially relevant when working with relatively scarce or sensitive data. The structure of GAN is represented by two neural networks: a generator G_{NN} and a discriminator D_{NN}. The generator takes a random variable $z \sim p_z(z)$ from a latent probability distribution $p_z(z)$. This is typically chosen as a uniform (or normal) distribution for $z \in (-1, 1)$. Using a composition $g_L \circ ... \circ g_2 \circ g_1(z)$ of (nonlinear) operations $\{g_i\}_{i=1}^{L}$ such that the generator prepares a fake sample $G_{\mathrm{NN}}(z)$ from the generator's probability distribution p_G, $G_{\mathrm{NN}}(z) \sim p_G(G_{\mathrm{NN}}(z))$. The goal is to make samples $\{G_{\mathrm{NN}}(z)\}_{s=1}^{N_{\mathrm{s}}}$, as close to the training dataset as possible, in terms of their sample distributions. If true samples $x \sim p_{\mathrm{data}}(x)$ are drawn from a (generally unknown) probability distribution $p_{\mathrm{data}}(x)$, our goal is to match $p_G(G_{\mathrm{NN}}(z)) \approx p_{\mathrm{data}}(x)$. This is achieved by training the discriminator network D_{NN} to distinguish true from fake samples, while improving the quality of generated samples $\{G_{\mathrm{NN}}(z)\}_{s=1}^{N_{\mathrm{s}}}$, optimizing a minimax loss

$$\min_{G_{\mathrm{NN}}} \max_{D_{\mathrm{NN}}} \mathcal{L}_{\mathrm{GAN}} = \min_{G_{\mathrm{NN}}} \max_{D_{\mathrm{NN}}} \Big\{ \mathbb{E}_{x\sim p_{\mathrm{data}}(x)}[\log D_{\mathrm{NN}}(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D_{\mathrm{NN}}(G_{\mathrm{NN}}(z)))] \Big\}, \tag{11}$$

where D_{NN} and G_{NN} are the trainable functions represented by the discriminator and generator, respectively. The first loss term in Eq. (11) represents the log-likelihood maximization that takes a *true* sample from the available dataset, and maximizes the probability for producing these samples by adjusting variational parameters. The second term trains G_{NN} to minimize the chance of being caught by the discriminator. Most importantly, we note that $G_{\mathrm{NN}}(z)$ is a function that converts a random sample $z \sim \texttt{uniform}(-1, 1)$ into a sample from the trained GAN distribution — therefore representing a *quantile-like function*. This is the connection we develop further using the qGAN training.

Quantum GANs follow the same ideology as their classical counterparts, but substitute the neural representation of the generator G_{NN} and/or discriminator D_{NN} by quantum neural networks. In the following, we denote them as G_{Q} and D_{Q}, respectively. The schedule of qGAN training and circuits are presented in Fig. 7. To apply qGANs for the same task of OU process learning, we concentrate on a continuous qGAN that uses the feature map encoding [66, 71]. We follow the training strategy from Ref. [66]. We try to model the normal distribution with zero mean and standard deviation of 0.2. Both the discriminator and generator use $N = 6$ registers with the expressive Chebyshev tower feature map [45] followed by $d = 6$

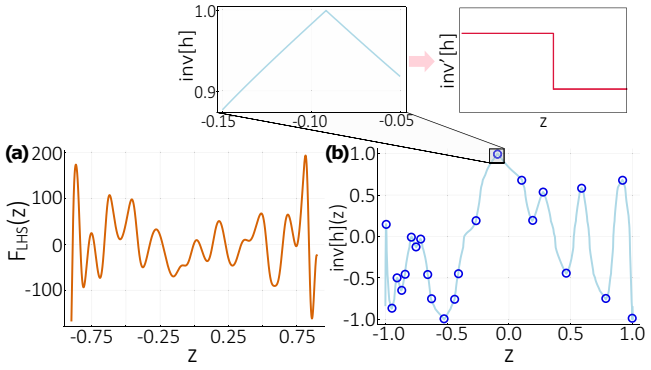

FIG. 9: **qGAN quantile function analysis.** In **(a)** we plot the LHS of Eq. (13) for the trained qGAN generator $G_Q(z)$ shown in Fig. 8(a). The resulting function is oscillatoric but smooth. **(b)** Inverse mapping function inv$[h]$ shown as a function of z. It transforms $G_Q(z)$ into the increasing quantile function $Q_{GAN}(\tilde{z})$ that is plotted in Fig. 8(d). We highlight the non-differentiable points by blue circles, and zoom in on the characteristic behavior in the inset above. The derivative is not defined at the discontinuity (top right inset).

HEA ansatz. The readout for the generator uses $\langle \hat{Z}_1 \rangle$ expectation, and the discriminator has the cost function such that we readout $(\langle \hat{Z}_1 \rangle + 1)/2 \in [0, 1]$ modeling the probability. As before, we use `Adam` and train the qGAN for 2000 epochs using the loss function (11). Due to the minimax nature of the training, the loss oscillates and instead of reaching (global) optimum qGAN tries to reach the Nash equilibrium. Unlike QQM training, we cannot simply use variational parameters for the final epoch, and instead test the quality throughout. To get the highest quality generator we test how close together are the discriminator (L_D) and generator (L_G) loss terms. If they are within $\epsilon = 0.1$ distance we perform the Kolmogorov-Smirnov (KS) test [59] and check the distance between the currently generated samples and the training dataset. The result with minimal KS is chosen. We stress that KS is not used for training, and is exclusively for choosing the best result.

The results for qGAN training are shown in Fig. 8. The trained generator $G_Q(z)$ is shown as a function of the latent variable z. We note that it has a strongly-oscillating nature. The training is shown separately for the generator (blue curve) and discriminator (red curve) loss terms. They oscillate around the analytic value for the Nash equilibrium (black dotted line, `NE`), and briefly settle around NE after 1600 epochs where optimal circuit parameters are saved. We sample the qGAN generator using $N_s = 100,000$ and plot the normalized histogram in Fig. 8(c). We observe that the distribution roughly matches the target (solid curve, PDF), though finer points are missing [cf. Fig. 6(c)], including the missing tail at negative values. Naturally, the generator of qGAN $G_Q(z)$ does the same job as the trained quantile function $G(z)$ from previous subsections. We proceed to connect the two explicitly.

Reordered quantile functions and their DEs. The main difference between the quantile function and the generator of qGAN is that the true QF is a strictly monotonically increasing function, while the qGAN generator G_Q is not. We can connect them by noticing that qGAN works with the latent variable $z \in \mathcal{Z}$, which we can rearrange into a QF by ordering the observations and assigning them the *ordered* latent variable $\tilde{z} \in \tilde{\mathcal{Z}}$ (both functions produce the same sample distribution). It is convenient to define a mapping $h : \tilde{\mathcal{Z}} \to \mathcal{Z}$ for G_Q which rearranges it into increasing form, $Q_{GAN}(\tilde{z}) = G_Q(h(\tilde{z}))$. In practice, finding $h(\tilde{z})$ requires the evaluation of $G_Q(z)\ \forall\ z \in \mathcal{Z}$ and re-assigning the samples to values of $\tilde{\mathcal{Z}}$ in ascending order. Importantly, both h and its inverse inv$[h] : \mathcal{Z} \to \tilde{\mathcal{Z}}$ can be defined in this process. In Fig. 8(d) we show the results of reordering for the generator function in Fig. 8(a). The reordered quantile Q_{GAN} is plotted (dashed curve), approximately matching the target quantile (solid curve). We observe that the center of the quantile is relatively well approximated but the tails are not (particularly for $\tilde{z} < 0$). This agrees with what is observed in the sampling shown in Fig. 8(c). Having established the correspondence for qGAN-based generative modeling and quantile-based modeling we ask the question: can we apply differential equations to the quantile-like function to add differential constraints, and evolve the system in time enabling generative modelling?

DISCUSSION

The answer to the question above is far from trivial. To use a re-ordered qGAN quantile function for further training and time-series generation we need to account for the mapping when writing differential equations of quantile mechanics. Let us look into a specific case to develop an intuition on the behavior of re-ordered quantile functions with differential equations. A quantile function $Q(\tilde{z})$ of a normal distribution with mean μ and standard deviation σ satisfies a quantile ODE [79, 80]

$$\frac{d^2Q}{d\tilde{z}^2} - \frac{Q-\mu}{\sigma^2}\left(\frac{dQ}{d\tilde{z}}\right)^2 = 0, \tag{12}$$

where we use the tilde notation $\tilde{z}$ to highlight that this is an ordered variable. Assuming perfect training such that $Q_{GAN}(\tilde{z}) = G_Q(h(\tilde{z}))$ closely matches $Q(\tilde{z})$, we substitute it into Eq. (12), and observe that the original qGAN generator obeys

$$\frac{d^2G_Q(z)}{dz^2} - \frac{G_Q(z)-\mu}{\sigma^2}\left(\frac{dG_Q(z)}{dz}\right)^2 = \frac{\text{inv}[h]''(z)}{\text{inv}[h]'(z)}\frac{dG_Q(z)}{dz}. \tag{13}$$

The left-hand side (LHS) of Eq. (13) has the same form as for the true QF [cf. Eq (12)], but the right-hand side (RHS) differs from zero and involves derivatives of the inverted mapping function inv$[h](z)$. This has important implications for training $G_Q(z)$ with differential constraints, as the loss term includes the difference between LHS and RHS. Let us analyze the example of the quantile ODE in Eq. (13). In Fig. 8(a) we plot the LHS for $G_Q(z)$ coming from the qGAN training. The result is a smooth function, and we expect all relevant terms, including derivatives $dG_Q(z)/dz$ or $d^2G_Q(z)/dz^2$, can be evaluated and trained at all points of the latent space. However, the problem arises when RHS enters the picture. The

additional term strongly depends on the contributions coming from inverse map derivatives $\mathrm{inv}[h]'$ and $\mathrm{inv}[h]''$. At the same time we find that the map from a non-monotonic to a monotonically increasing function is based on a multivalued function (see discussion and examples in Supplemental Information). Furthermore the inverse of the map (along with the map itself) is continuous but not smooth — it becomes non-differentiable at some points due to $G_{\mathrm{Q}}(z)$ oscillations. As an example in Fig. 9(b) we show $\mathrm{inv}[h]$ from training in Fig. 8, highlighting the points with non-analytic behavior blue circles. The inset for Fig. 9(b) clearly shows the discontinuity. This translates to the absence of $\mathrm{inv}[h]'(z)$ at a set of points, which unlike zero derivatives cannot be removed by reshuffling the terms in the loss function. The discovered unlikely property of the mapping puts in jeopardy the attempts to use differential-based learning for qGAN generators. While more studies are needed to estimate the severity of discontinuities (and if the set of such points can be excluded to yield stable training), our interim conclusion is that quantile functions in the canonical increasing form are more suitable for evolution and time series generation.

Closing notes. We proposed a distinct quantum algorithm for generative modelling from stochastic differential equations. Summarizing the findings, we have developed the understanding of generative modelling from stochastic differential equations based on the concept of quantile functions. We proposed to represent the quantile function with a trainable (neural) representation, which may be classical- or quantum-based. In particular, we focused on parameterizing the trainable quantile function with a differentiable quantum circuit that can learn from data and evolve in time as governed by quantile mechanics equations. Using Ornstein-Uhlenbeck as an example, we benchmark our approach and show that it gives a robust strategy for generative modelling in the NISQ setting. Furthermore, we notice that adversarial schemes as continuous qGAN lead to modified quantile-like function that potentially have intrinsic obstacles for evolving them in time. We conclude by saying that the strategy we propose uses the large expressive power of quantum neural networks, and we expect elements of the approach can be used for other architectures.

METHODS

In this section we describe the details of circuit differentiation and the proposed boundary handling procedure for multivariate functions.

Calculating second-order derivatives

For solving SDEs we need to access the derivatives of the circuits representing quantile functions dG/dz, d^2G/dz^2, where z is a latent variable. This can be done using automatic differentiation techniques for quantum circuits, where for near-term devices and specific gates we can use the parameter shift rule [85, 93, 94]. The differentiation of quantum feature maps follows the DQC strategy [45], where we estimate dG/dz as a sum of expectation values

$$\frac{dG(z)}{dz} = \frac{1}{2}\sum_{j=1}^{N} \varphi_j'(z)\Big(\langle G_j^+\rangle - \langle G_j^-\rangle\Big) \tag{14}$$

where $\langle G_j^+\rangle$ and $\langle G_j^-\rangle$ denote the evaluation of the circuit with the j-th gate parameter shifted positively and negatively by $\pi/2$. This generally requires $2N$ circuit evaluations. The (non-linear) function $\varphi_j(z)$ represents the z-dependent rotation phase for j-th qubit. A popular choice is $\varphi_j(z) = \arcsin(z)$ (same for all qubits) referred as a product feature map, and other choices include tower feature maps [45].

Quantum circuit differentiation for higher-order derivatives was recently considered in several studies [94–96]. Extending the feature map differentiation to the second order, we use parameter shift rule alongside the product rule, and calculate d^2G/dz^2 as

$$\begin{aligned}\frac{d^2G(z)}{dz^2} &= \frac{1}{2}\sum_{j=1}^{N} \varphi_j''(z)\Big(\langle G_j^+\rangle - \langle G_j^-\rangle\Big) \\ &+ \frac{1}{4}\sum_{j=1}^{N}\sum_{k=1}^{N} \varphi_j'(z)\varphi_k'(z)\Big(\langle G_{jk}^{++}\rangle - \langle G_{jk}^{+-}\rangle - \langle G_{jk}^{-+}\rangle + \langle G_{jk}^{--}\rangle\Big),\end{aligned} \tag{15}$$

where $\langle G_{jk}^{++}\rangle$ ($\langle G_{jk}^{--}\rangle$) denotes the evaluation of the circuit with the rotation angles on qubits j and k are shifted positively (negatively) by $\pi/2$. Similarly, $\langle G_{jk}^{+-}\rangle$ ($\langle G_{jk}^{-+}\rangle$) are defined for shifts in opposite directions. If implemented naively, the derivative in Eq. (15) requires $2N + 4N^2$ evaluations of circuit expectation values. This can be reduced by making use of symmetries in the shifted expectation values and reusing/caching previously calculated values, i.e. those for the function and its first-order derivative. The reduced number of *additional* circuit evaluations required for the calculation of the second derivative is $2N^2$.

Boundary handling for PDEs

As we consider differential equations with more than one independent variable, we need to develop a strategy for implementing (handling) the boundary in this situation. Let us consider a function of n variables. We consider an initial condition of $f(t = 0, z) = u_0(z)$, where z is a vector of $n - 1$ independent variables, and the first variable usually corresponds to time. Here we extend several techniques, corresponding to pinned type and floating type boundary handling, previously considered for the single-variable case in Ref. [45].

When considering just one independent variable, a pinned boundary handling corresponds to encoding the function as $f(t) = G(t)$. The boundary is then 'pinned' into place by use of a boundary term in the loss function $\mathcal{L}^B = \Big[f(t_0) - u_0\Big]^2$. For multiple independent variables this generalizes to $f(t, z) = G(t, z)$ and $\mathcal{L}^B = \sum_i \Big[f(t_0, z_i) - u(z_i)\Big]^2$, where $\{z_i\}$ are the set of points along $t = 0$ at which the boundary is being pinned.

When using the floating boundary handling the boundary is implemented during the function encoding. In this case the function encoding is generalized from its single-variable representation, $f(t) = u_0 - G(0) + G(t)$, to the multivariate case as

$$f(t, z) = u_0(z) - G(0, z) + G(t, z). \tag{16}$$

This approach does not require the circuit-embedded boundary, but instead needs derivatives of $u_0(z)$ for calculating the derivatives of f with respect to any $z \in \mathcal{Z}$.

[1] C. W. Gardiner and P. Zoller, *Quantum Noise* (Springer-Verlag, Berlin/Heidelberg, 2004).

[2] H.-P. Breuer and F. Petruccione, *The Theory of Open Quantum Systems* (Oxford, 2002).

[3] N. G. Van Kampen, *Stochastic differential equations*, Physics Reports **24**, 171 (1976).

[4] B. Leimkuhler and C. Matthews, *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods* (Springer, 2015).

[5] D. D. Holm, *Variational principles for stochastic fluid dynamics*, Proc. R. Soc. A **471**, 20140963 (2015).

[6] C. Cintolesi and E. Mémin, *Stochastic Modelling of Turbulent Flows for Numerical Simulations*, Fluids **5**, 108 (2020).

[7] U. Dobramysl, M. Mobilia, M. Pleimling, and U. C. Täuber, *Stochastic population dynamics in spatially extended predator-prey systems*, J. Phys. A: Math. Theor. **51**, 063001 (2018).

[8] L. J. S. Allen, *An Introduction to Stochastic Epidemic Models.* (2008) In: Brauer F., van den Driessche P., Wu J. (eds) Mathematical Epidemiology. Lecture Notes in Mathematics, vol 1945. (Springer, Berlin, Heidelberg, 2008).

[9] L. J. S. Allen, *A primer on stochastic epidemic models: Formulation, numerical simulation, and analysis*, Infect. Dis. Model. **2**, 128 (2017).

[10] Y. Rajabzadeh, A. H. Rezaie, and H. Amindavar, *A dynamic modeling approach for anomaly detection using stochastic differential equations*, Digit. Signal Process.**54**, 1 (2016).

[11] B. Øksendal, *Stochastic Differential Equations: An Introduction with Applications* (Springer-Verlag, Berlin/Heidelberg, 1998).

[12] T. Leung and Li Xin, *Optimal Mean Reversion Trading: Mathematical Analysis and Practical Applications* (World Scientific, 2016).

[13] V. I. Bogachev, N. V. Krylov, M. Röckner, and S. V. Shaposhnikov, *Fokker–Planck–Kolmogorov Equations* (Mathematical surveys and monographs, volume 207, 2015).

[14] R. T. Q. Chen and D. Duvenaud, *Neural Networks with Cheap Differential Operators*, arXiv:1912.03579 [cs.LG] (2019).

[15] Xuechen Li, Ting-Kam L. Wong, R. T. Q. Chen, and D. K. Duvenaud, *Scalable Gradients and Variational Inference for Stochastic Differential Equations*, Proceedings of The 2nd Symposium on Advances in Approximate Bayesian Inference, PMLR **118**, 1-28, (2020).

[16] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations* (Springer, Berlin, 1992).

[17] Dongkun Zhang, Ling Guo and G. E. Karniadakis, *Learning in Modal Space: Solving Time-Dependent Stochastic PDEs Using Physics-Informed Neural Networks*, arXiv:1905.01205 [cs.LG] (2019).

[18] Dongkun Zhang, Lu Lu, Ling Guo, and G. E. Karniadakis, *Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems*, J. Comp. Phys. **397**, 108850 (2019).

[19] M. A. Nabian and H. Meidani, *A deep learning solution approach for high-dimensional random differential equations*, Prob. Eng. Mech. **57**, 14 (2019).

[20] L. Yang, D. Zhang and G. E. Karniadakis, *Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations*, arXiv:1811.02033 [stat.ML] (2018).

[21] P. Kidger, J. Foster, Xuechen Li, H. Oberhauser, and T. Lyons, *Neural SDEs as Infinite-Dimensional GANs*, arXiv:2102.03657 [cs.LG] (2021).

[22] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Networks*, arXiv:1406.2661 [stat.ML] (2014).

[23] S. L. Brunton, J. L. Proctor, J. N. Kutz, and W. Bialek, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, PNAS **113**, 3932 (2016).

[24] H. Schaeffer, *Learning partial differential equations via data discovery and sparse optimization*, Proc. R. Soc. A **473**, 20160446 (2017).

[25] M. Raissi, *Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations*, arXiv:1801.06637 [stat.ML] (2018).

[26] M. Maslyaev, A. Hvatov, and A. Kalyuzhnaya, *Data-driven PDE discovery with evolutionary approach*, Proc. ICCS 2019 **11540**, 635 (2019); arXiv:1903.08011 [cs.NE] (2019).

[27] P. A. K. Reinbold and R. O. Grigoriev, *Data-driven discovery of partial differential equation models with latent variables*, Phys. Rev. E. **100**, 1 (2019).

[28] S. Aaronson and A. Arkhipov, *The Computational Complexity of Linear Optics*, Theory of Computing. **9**, 143 (2013).

[29] A. P. Lund, M. J. Bremner, and T. C. Ralph, *Quantum sampling problems, BosonSampling and quantum supremacy*, npj Quantum Information **3**, 15 (2017).

[30] J. M. Arrazola, P. Rebentrost, and C. Weedbrook, *Quantum supremacy and high-dimensional integration*, arXiv:1712.07288 [quant-ph] (2017).

[31] A. Deshpande, A. Mehta, T. Vincent, N. Quesada, M. Hinsche, M. Ioannou, L. Madsen, J. Lavoie, Haoyu Qi, J. Eisert, D. Hangleiter, B. Fefferman, and Ish Dhand, *Quantum Computational Supremacy via High-Dimensional Gaussian Boson Sampling*, arXiv:2102.12474 [quant-ph] (2021).

[32] F. Arute et al., *Quantum supremacy using a programmable superconducting processor*, Nature **574**, 505 (2019).

[33] Han-Sen Zhong et al., *Quantum computational advantage using photons*, Science **370**, 1460 (2020).

[34] S. Chakrabarti, R. Krishnakumar, G. Mazzola, N. Stamatopoulos, S. Woerner, and W. J. Zeng, *A Threshold for Quantum Advantage in Derivative Pricing*, Quantum **5**, 463 (2021).

[35] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, *Quantum Computing for Finance: State-of-the-Art and Future Prospects*, IEEE Transactions on Quantum Engineering **1**, 3101724 (2020).

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

[37] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, *Parameterized quantum circuits as machine learning models*, Quantum Science and Technology **4**, 043001 (2019).

[38] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, Xiao Yuan, L. Cincio, and P. J. Coles, Variational Quantum Algorithms, arXiv:2012.09265 [quant-ph] (2020).

[39] M. Schuld and N. Killoran, *Quantum Machine Learning in Feature Hilbert Spaces*, Phys. Rev. Lett. **122**, 040504 (2019).

[40] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, *Transfer learning in hybrid classical-quantum neural networks*, Quantum **4**, 340 (2020).

[41] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner *The power of quantum neural networks*, Nat. Com-

put. Sci. **1**, 403 (2021).
[42] Samuel Yen-Chi Chen and Shinjae Yoo, *Federated Quantum Machine Learning*, Entropy **23**, 460 (2021).
[43] Samuel Yen-Chi Chen, Tzu-Chieh Wei, Chao Zhang, Haiwang Yu, Shinjae Yoo, *Quantum Convolutional Neural Networks for High Energy Physics Data Analysis*, arXiv:2012.12177 [cs.LG] (2020).
[44] M. Lubasch, Jaewoo Joo, P. Moinier, M. Kiffner, and D. Jaksch, *Variational quantum algorithms for nonlinear problems*, Phys. Rev. A 101, 010301(R) (2020).
[45] O. Kyriienko, A. E. Paine, and V. E. Elfving, *Solving nonlinear differential equations with differentiable quantum circuits*, Phys. Rev. A **103**, 052416 (2021).
[46] M. Knudsen and C. B. Mendl, *Solving Differential Equations via Continuous-Variable Quantum Computers*, arXiv:2012.12220 [quant-ph]
[47] P. Garcia-Molina, J. Rodriguez-Mediavilla, and J. J. Garcia-Ripoll, *Solving partial differential equations in quantum computers*, arXiv:2104.02668 [quant-ph]
[48] P. Rebentrost, B. Gupt, and T. R. Bromley, *Quantum computational finance: Monte Carlo pricing of financial derivatives*, Phys. Rev. A **98**, 022321 (2018).
[49] N. Stamatopoulos, D. J. Egger, Yue Sun, C. Zoufal, R. Iten, Ning Shen, and Stefan Woerner, Option Pricing using Quantum Computers, Quantum **4**, 291 (2020).
[50] K. Kubo, Y. O. Nakagawa, S. Endo, and S. Nagayama, *Variational quantum simulations of stochastic differential equations*, Phys. Rev. A **103**, 052425 (2021).
[51] J. Gonzalez-Conde, Ángel Rodríguez-Rozas, Enrique Solano, Mikel Sanz, *Pricing Financial Derivatives with Exponential Quantum Speedup*, arXiv:2101.04023 [quant-ph]
[52] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, Jieping Ye, *A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications*, arXiv:2001.06937 [cs.LG] (2020).
[53] S. Lloyd and C. Weedbrook, *Quantum Generative Adversarial Learning*, Phys. Rev. Lett. **121**, 040502 (2018).
[54] P.-L. Dallaire-Demers and N. Killoran, *Quantum generative adversarial networks*, Phys. Rev. A **98**, 012324 (2018).
[55] Jin-Guo Liu and Lei Wang, *Differentiable Learning of Quantum Circuit Born Machines*, Phys. Rev. A **98**, 062324 (2018).
[56] Jinfeng Zeng, Yufeng Wu, Jin-Guo Liu, Lei Wang, and Jiangping Hu, *Learning and inference on generative adversarial quantum circuits*, Phys. Rev. A **99**, 052306 (2019).
[57] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, A. Perdomo-Ortiz, *A generative modeling approach for benchmarking and training shallow quantum circuits*, npj Quantum Information **5**, 45 (2019).
[58] Yuxuan Du, Min-Hsiu Hsieh, Dacheng Tao, *Efficient Online Quantum Generative Adversarial Learning Algorithms with Applications*, arXiv:1904.09602 [quant-ph] (2019).
[59] C. Zoufal, A. Lucchi, and S. Woerner, *Quantum Generative Adversarial Networks for learning and loading random distributions*, npj Quantum Information **5**, 45 (2019).
[60] Haozhen Situ, Zhimin He, Yuyi Wang, Lvzhou Li, and Shenggen Zheng, *Quantum generative adversarial network for generating discrete distribution*, Information Sciences **538**, 193 (2020).
[61] Su Yeon Chang, S. Herbert, S. Vallecorsa, E. F. Combarro, and R. Duncan, *Dual-Parameterized Quantum Circuit GAN Model in High Energy Physics*, arXiv:2103.15470 [quant-ph] (2021).
[62] M. Y. Niu, A. Zlokapa, M. Broughton, S. Boixo, M. Mohseni, V. Smelyanskyi, H. Neven, *Entangling Quantum Generative Adversarial Networks*, arXiv:2105.00080 [quant-ph] (2021).
[63] M. Benedetti, E. Grant, L. Wossnig, and S. Severini, *Adversarial quantum circuit learning for pure state approximation*, New J. Phys. **21**, 043023 (2019).
[64] P. Braccia, F. Caruso, and L. Banchi, *How to enhance quantum generative adversarial learning of noisy information*, New J. Phys. **23**, 053024 (2021).
[65] He-Liang Huang et al., *Experimental Quantum Generative Adversarial Networks for Image Generation*, arXiv:2010.06201 [quant-ph] (2020).
[66] J. Romero and A. Aspuru-Guzik, *Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions*, Adv. Quantum Technol. **4**, 2000003 (2021).
[67] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, *Cost function dependent barren plateaus in shallow parametrized quantum circuits*, Nature Commun. **12**, 1791 (2021).
[68] B. Coyle, D. Mills, V. Danos, and E. Kashefi, *The Born Supremacy: Quantum Advantage and Training of an Ising Born Machine*, npj Quantum Inf **6**, 60 (2020).
[69] B. Coyle, M. Henderson, Justin Chan Jin Le, N. Kumar, M. Paini, and E. Kashefi, *Quantum versus classical generative modelling in finance*, Quantum Sci. Technol. **6**, 024013 (2021).
[70] A. Kondratyev, *Non-Differentiable Learning of Quantum Circuit Born Machine with Genetic Algorithm*, (SSRN, 2020).
[71] A. Anand, J. Romero, M. Degroote, and A. Aspuru-Guzik, *Noise Robustness and Experimental Demonstration of a Quantum Generative Adversarial Network for Continuous Distributions*, Adv. Quantum Technol. **4**, 2000069 (2021).
[72] Takahiro Goto, Quoc Hoan Tran, and Kohei Nakajima, *Universal Approximation Property of Quantum Feature Map*, arXiv:2009.00298 [quant-ph] (2020).
[73] Tong Li, Shibin Zhang, and Jinyue Xia, *Quantum Generative Adversarial Network: A Survey*, Computers, Materials & Continua **64**,401 (2020).
[74] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, *Quantum Boltzmann Machine*, Phys. Rev. X **8**, 021050 (2018).
[75] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, Dacheng Tao, *Expressive Power of Parameterized Quantum Circuits*, Phys. Rev. Research **2**, 033125 (2020).
[76] J. Liu, Ke-Thia Yao, and F. Spedalieri, *Dynamic Topology Reconfiguration of Boltzmann Machines on Quantum Annealers*, Entropy **22**, 1202 (2020).
[77] Y. Takaki, K. Mitarai, M. Negoro, K. Fujii, and M. Kitagawa, *Learning temporal data with a variational quantum recurrent neural network*, Phys. Rev. A **103**, 052414 (2021).
[78] J. P. Boyd, *Solving transcendental equations: The Chebyshev polynomial proxy and other numerical rootfinders, perturbation series, and oracles* (Society for Industrial and Applied Mathematics, Philadelphia, 2014).
[79] G. Steinbrecher and W. T. Shaw, *Quantile mechanics*, European Journal of Applied Mathematics **19**, 87 (2008).
[80] J. A. Carillo and G. Toscani, *Wasserstein metric and large-time asymptotics of nonlinear diffusion equations*, in New Trends in Mathematical Physics, 234–244 (World Sci. Publ., Hackensack, NJ, 2004).
[81] W. T. Shaw, *Sampling Student's T distribution – use of the inverse cumulative distribution function*, J. Comput. Fin. **9**, 37 (2006).
[82] W. T. Shaw, T. Luu, and N. Brickman, *Quantile Mechanics II: Changes of Variables in Monte Carlo methods and GPU-Optimized Normal Quantiles*, European Journal of Applied Mathematics **25**, 177 (2014).

[83] W. Gilchrist, *Statistical Modelling with Quantile Functions* (CRC Press, London, 2000).

[84] K. Hornik, M. Tinchcombe, H. White, *Multilayer Feedforward Networks are Universal Approximators* Neur. Netw. **2** 359-366 (1998).

[85] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, *Quantum circuit learning*, Phys. Rev. A **98**, 032309 (2018).

[86] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets*, Nature **549**, 242 (2017).

[87] O. Vasicek, *An equilibrium characterization of the term structure*, Journal of Financial Economics **5**, 177 (1977).

[88] R. S. Mamon, *Three Ways to Solve for Bond Prices in the Vasicek Model*, Advances in Decision Sciences **8**, 131526 (2004).

[89] J. Hull and A. White, *Pricing Interest-Rate-Derivative Securities*, Rev. Financ. Stud. **3**, 573 (1990).

[90] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* (United States. Government Printing Office., Washington D.C., 1972).

[91] Xiu-Zhe Luo, Jin-Guo Liu, Pan Zhang, Lei Wang, *Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design*, Quantum **4**, 341 (2020).

[92] C. Rackauckas and Qing Nie, *DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia*, J. Open Res. Softw. **5**, 15 (2017).

[93] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, *Evaluating analytic gradients on quantum hardware*, Phys. Rev. A **99**, 032331 (2019).

[94] K. Mitarai, Y. O. Nakagawa, and W. Mizukami, *Theory of analytical energy derivatives for the variational quantum eigensolver*, Phys. Rev. Res. **2**, 013129 (2020).

[95] M. Cerezo and P. J. Coles, *Higher order derivatives of quantum neural networks with barren plateaus*, Quantum Sci. Technol. **6**, 035006 (2021).

[96] A. Mari, T. R. Bromley, and N. Killoran, *Estimating the gradient and higher-order derivatives on quantum hardware*, Phys. Rev. A **103**, 012405 (2021).

[97] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, S. Ganguli *Deep Unsupervised Learning using Nonequilibrium Thermodynamics* arXiv:1503.03585 [cs.LG] (2015).

[98] J. Ho, A. Jain, P. Abbeel *Denoising Diffusion Probabilistic Models* arXiv:2006.11239 [cs.LG] (2021).

[99] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, B. Poole, *Score-Based Generative Modeling through Stochastic Differential Equations*, arXiv:2011.13456 [cs.LG] (2020).

[100] B.D.O. Anderson, *Reverse-time diffusion equation models* Stoch. Proc. Appl. **12** 3 (1982).

[101] T. Yan, H. Zhang, T. Zhou, Y. Zhan, Y. Xia, *ScoreGrad: Multivariate Probabilistic Time Series Forecasting with Continuous Energy-based Generative Models*, arXiv:2106.10121 [cs.LG] (2021).

Supplemental Information

Analytic quantile reordering for qGAN generators

To understand the reordering procedure for qGAN generators and strictly increasing quantile functions, let us consider a simple example.

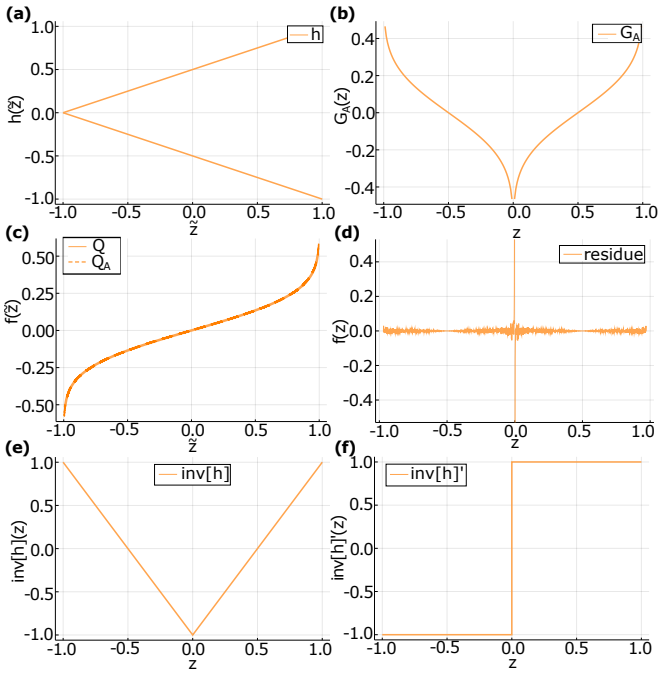

FIG. 10: **qGAN quantile reordering example.** **(a)** Mapping function $h(\tilde{z}) = \pm(\tilde{z}+1)/2$. **(b)** Non-ordered quantile function $G_{\rm A}(z) = Q(\text{inv}[h](z))$ corresponding to mapping in **(a)** and normal quantile function. **(c)** Known quantile function (solid curve, Q) compared to numerical ordering of G_A (dashed line, Q_A). **(d)** RHS of Eq. (15) in the main text, plotted for $G_{\rm A}$ shown in **(b)**. Derivatives of inv$[h]$ are calculated using finite differencing. **(e)** Inverse mapping function inv$[h]$ plotted for different values of the latent variable z. We note the point of non-differentiability at $z = 0$. **(f)** Analytic first derivative of inv$[h]$ that has a discontinuity at $z = 0$.

We start by considering a quantile-like generator $G_{\rm A}(z)$ and use the mapping h that reorders it into ideal quantile function (QF) for the normal distribution. The mapping reads $h(\tilde{z}) = \pm\left(\tilde{z}+1\right)/2$, and is shown in Fig. 10(a) as a multivalued function. It ensures that if we start from the normal QF, we arrive to $G_{\rm A}(z)$ with a single dip. The corresponding qGAN-like generator $G_{\rm A}(z)$ with a single dip is shown in Fig. 10(b) (we consider $\mu = 0$ and $\sigma = 0.2$). Our motivation is to understand how the presence of nonmonotonicity changes the behaviour of the system.

First, let us check that the reordering into increasing quantile function works as expected. Assigning the values of $G_{\rm A}(z)$ in the ascending order we get the reordered QF $Q_{\rm A}(\tilde{z})$. This is plotted in Fig. 10(c), matching the ideal $Q(\tilde{z})$ as expected. Once we have established the mapping for ideal reordering, let us look at its properties. In the results section we discussed how the reordered quantile function from qGAN training matches the appropriate quantile ODE. We can perform the same check for the simple re-ordering presented above. Evaluating the difference between the RHS and LHS of Eq. (13) (akin to loss term) for $G_{\rm A}(z)$ and known mapping $h(\tilde{z})$, we observe that the difference remains zero everywhere (we have a perfect solution), apart from the middle point $z = 0$ where it diverges [Fig. 10(d)]. For calculating the derivatives of inv$[h]$ we use finite differencing (Euler's method), similarly for the qGAN case, thus observing small noise coming from numerical differentiation. The reason behind the unfavorable loss term behavior can be tracked to the properties of the mapping function. We show the inverse mapping inv$[h]$ plotted in Fig. 10(e), and its derivative inv$[h]'$ is presented in Fig. 10(f). We see that inv$[h]$ has a point of non-differentiability at $z = 0$ and inv$[h]'$ is discontinuous there. This provides the intuition behind the divergence. When training the DE-based loss for Eq. (13) with $z = 0$ included the loss becomes non-trainable. We stress that the same is observed for the non-ideal $G_{\rm Q}$, where multiple non-differentiable points appear that we do not know in advance. The issue of developing efficient workflow for training $G_{\rm Q}(z)$, also including the time dimension, remains an important area for the future research.

Time evolution for data-inferred quantile function

In the Results section of the main text we detail how DQC is used to time evolve an analytic initial condition and how to learn the initial condition based on observations. We can also time evolve with DQC based on the observed initial quantile. The set up of the DQC is the same as when an analytic initial condition leading to Fig. 4. The result of the training is shown in Fig. 11, showing that the same quality of propagation is obtained.

Solving reverse-time Stochastic Differential Equations with QQM

Interesting connections between thermodynamics, machine learning and image synthesis have been uncovered in recent years. Recently, Denoising Diffusion Probabilistic Models (DDPM) [97] were shown to perform high quality image synthesis at state-of-the-art levels [98], sometimes better than other generative methods like GAN-approaches. Ref. [99] realized that such discrete DDPMs can also be modelled as continuous processes using *reverse-time* SDEs when combined with ideas from score-based generative modeling.

Ref. [100] derived the reverse-time form of general stochastic differential equations, being

$$d\boldsymbol{X}_t = \bar{f}(\boldsymbol{X}_t, t)dt + g(\boldsymbol{X}_t, t)dW_t, \tag{17}$$

where now a *modified* diffusion term $\bar{f}$ is given by

$$\bar{f}(\boldsymbol{X}_t, t) = f(\boldsymbol{X}_t, t) - g^2(\boldsymbol{X}_t, t)\nabla_{\boldsymbol{X}} \log\left[p(\boldsymbol{x}, t)\right] \tag{18}$$

Ref. [99] proposed to solve such equations with general-purpose numerical methods such as Euler-Maruyama and stochastic Runge-Kutta methods, as well as using predictor-corrector samplers. We envisage solving the reverse-time Fokker-Planck equation corresponding to Eq. (17) instead, and more precisely its quantilized form, using the method described in this paper. We note that the reverse-time form (not to be confused with backward-Kolmogorov) actually looks the

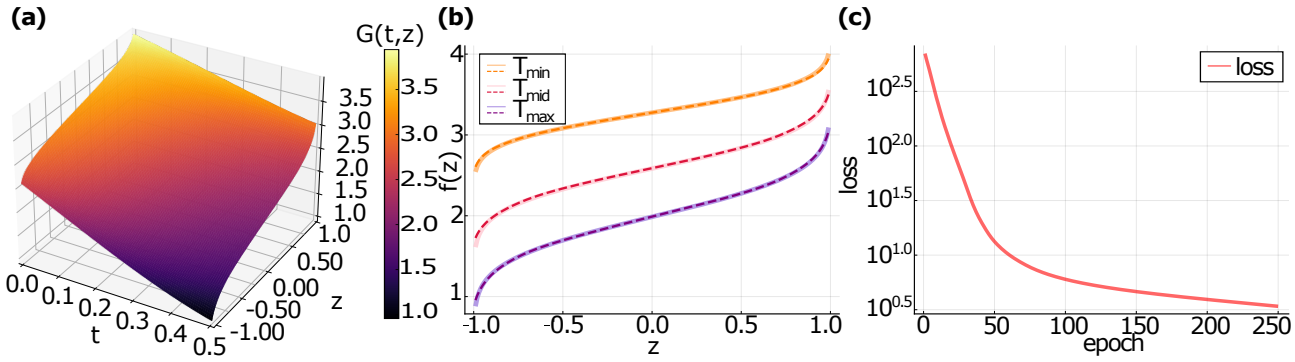

FIG. 11: **Time-evolution of Ornstein-Uhlenbeck process with data-inferred initial condition.** The initial conditions is taken from Fig. 6 and we use the same OU model parameters. **(a)** Surface plot for the quantile function evolved in time. **(b)** Quantile functions shown at three time points (being the same as in the main text). **(c)** Loss as a function of epoch from training in **(a)** and **(b)**.

same, but is simply solved backwards in time starting from a 'final condition' rather than an 'initial condition' data set.

As noted in Ref. [99], DDPM can be regarded as the discrete form of a stochastic differential equation (SDE). In Ref. [101], a general framework based on continuous energy-based generative models for time series forecasting is established. The training process at each step is composed of a time series feature extraction module and a conditional SDE based score matching module. The prediction can be achieved by solving reverse time SDE. The method is shown to achieve state-of-the-art results for multivariate time series forecasting on real-world datasets. These works imply that the method described here can be used for (multivariate) time-series forecasting and high-quality image synthesis.