Quantitative Finance

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/rquf20

Quantum-inspired variational algorithms for partial differential equations: application to financial derivative pricing

Tianchen Zhao, Chuhao Sun, Asaf Cohen, James Stokes & Shravan Veerapaneni

To cite this article: Tianchen Zhao, Chuhao Sun, Asaf Cohen, James Stokes & Shravan Veerapaneni (2024) Quantum-inspired variational algorithms for partial differential equations: application to financial derivative pricing, Quantitative Finance, 24:1, 1-11, DOI: 10.1080/14697688.2023.2259954

To link to this article: https://doi.org/10.1080/14697688.2023.2259954

Published online: 29 Sep 2023.

Submit your article to this journal

Article views: 296

View related articles

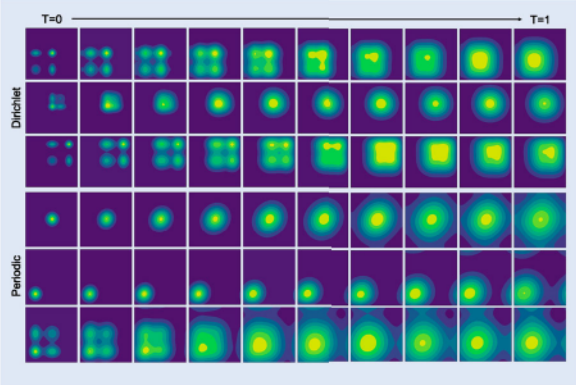
View Crossmark data

Citing articles: 1 View citing articles

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=rquf20

Quantitative Finance, 2024
Vol. 24, No. 1, 1–11, https://doi.org/10.1080/14697688.2023.2259954

Routledge
Taylor & Francis Group
Check for updates



Quantum-inspired variational algorithms for partial differential equations: application to financial derivative pricing

TIANCHEN ZHAO, CHUHAO SUN, ASAF COHEN *, JAMES STOKES and SHRAVAN VEERAPANENI

Department of Mathematics, University of Michigan, Ann Arbor, MI 48109, USA

(*Received 2 August 2022; accepted 1 September 2023; published online 29 September 2023*)

Variational quantum Monte Carlo (VMC) combined with neural-network quantum states offers a novel angle of attack on the curse-of-dimensionality encountered in a particular class of partial differential equations (PDEs); namely, the real- and imaginary time-dependent Schrödinger equation. In this paper, we present a simple generalization of VMC applicable to arbitrary time-dependent PDEs, showcasing the technique in the multi-asset Black-Scholes PDE for pricing European options contingent on many correlated underlying assets.

Keywords: Variational quantum algorithms; Variational quantum Monte Carlo; Multi-asset Black-Scholes PDE

1. Introduction

The field of stochastic variational algorithms has undergone dramatic recent developments based on two relatively recent, albeit independent, developments: (i) the availability of near-term quantum computers (Preskill 2018) and (ii) the existence of scalable stochastic algorithms for training deep neural networks. The former development has motivated a new research direction called variational quantum algorithms (VQAs) (Cerezo *et al.* 2021), in which the stochastic variational character of the algorithms render them suitable for noisy intermediate-scale quantum computers (Preskill 2018). The latter holds promise to accelerate a host of scientific computing problems including general-purpose solvers of partial differential equations (PDEs) using physics-informed neural-networks (PINNs) (Raissi *et al.* 2019) and has already made substantial headway in solving the time-(in)dependent Schrödinger equation in high dimensions using variational quantum Monte Carlo (VMC) with neural-network quantum states (Carleo and Troyer 2017). It is noteworthy that there exist many shared parallels between the fields of VQAs, VMC (Stokes *et al.* 2020, 2022) and PINNs. In particular, both VMC and VQAs hinge on the concept of adaptive stochastic estimation of the time-independent and time-dependent variational principles originally due to Rayleigh-Ritz and McLachlan (McLachlan 1964), respectively. VQAs and VMC differ essentially in the choice of parametrized quantum state and the associated adaptive sampling strategy used to estimate quantum expectation values. PINNs can be viewed as an alternative approach to McLachlan's variational principle. They gained significant traction outside the quantum physics literature and have been advocated for solving general PDEs that appear in other areas of science and engineering. This approach involves solving a non-convex optimization problem for the PDE residual in order to determine the space-time development of the state variable. A large number of variants of this proposal have been subsequently put forward targeting PDEs either in many spatial dimensions (e.g.Sirignano and

*Corresponding author. Email: shloshim@gmail.com

© 2023 Informa UK Limited, trading as Taylor & Francis Group

Spiliopoulos 2018, Wang *et al.* 2021) or parametric PDEs in low dimensions (e.g. Li *et al.* 2020). Deep backward stochastic differential equation methods (Weinan *et al.* 2017, Han *et al.* 2018, Beck *et al.* 2019, Huré *et al.* 2020) are another class of deep learning approaches that have been applied to parabolic PDEs that arise in mathematical finance.

In this paper, we introduce a generalization of McLachlan's variational principle applicable to a wide variety of time-dependent PDEs and propose a variational quantum Monte Carlo (VMC) stochastic approximate solution method utilizing autoregressive neural-network quantum states. A closely related algorithm has been recently introduced in the VQA literature† (Alghassi *et al.* 2021, Fontanela *et al.* 2021), which was motivated by the problem of solving linear PDEs using digital quantum computers without the necessity of HHL linear system solves (Harrow *et al.* 2009). Ref. Alghassi *et al.* (2021) argued for an approximate solution concept, with respect to which exponential quantum speedup could be achieved for state evolution, which however is overwhelmed by auxiliary costs of state preparation and extraction of properties of the state (see Alghassi *et al.* 2021, Section III.C for details). These complications of read-out and state preparation are not relevant to the VMC-based solver, however, since the VMC computing model permits efficient queries to arbitrary probability amplitudes. Interestingly, our generalization of McLachlan's variational principle is closely related to the neural Galerkin method recently put forward in Bruna *et al.* (2022), which also introduced a different neural-network-based stochastic solution method. One small difference compared to Bruna *et al.* (2022) is that we choose to work on a predefined mesh, which provides a simple method to implement non-trivial boundary conditions necessary for financial applications. The use of a mesh is not mandatory, however, and this paper paves the way toward a mesh-free generalization. Indeed, in the final stages of preparation of this article, Ref. Reh and Gärttner (2022) appeared, which proposes a mesh-free flow-based solution of probabilistic PDEs such as the Fokker-Planck equation. Unlike Reh and Gärttner (2022), however, our approach does not require that the state variable of the underlying PDE corresponds to a probability density.

The speedup obtainable by the approach advocated here has practical applications in overcoming the curse-of-dimensionality in high-dimensional PDEs, particularly in situations where fine-grained information about the state variable is required, such as gradients with respect to the independent variables. One such situation is quantum many-body physics, where kinetic energy observables depend on second-order spatial derivatives. Another example is the pricing and hedging of contingent claims in multi-asset financial markets. For example, if a Financial institution writes a call contract valued at $V(t)$ on an underlying asset of value $S(t)$ at time t, then according to the Black-Scholes framework the firm can minimize their exposure by buying a number of shares equal to $\Delta(t) = \partial V/\partial S$. The approach is applicable to general time-dependent PDEs expressible in first-order form, although we only consider the inhomogeneous linear case in the numerical experiments. Specifically, we chose to focus on the multi-asset Black-Scholes PDE for pricing European contingent claims because of its well-known relation with the time-dependent Schrödinger equation (TDSE) as well as its importance in computational finance. The techniques developed in this paper are not limited to the Black-Scholes framework, however, and can be straightforwardly generalized to pricing models with stochastic volatility, for example. In addition, the technique generalizes to many other linear PDEs in which the state variable does not admit a probabilistic interpretation.

The organization of the paper is as follows: In section 2 we generalize the McLachlan variational principle to general time-dependent PDEs in a model-agnostic manner, which is applicable in the purely classical or quantum setting. In section 3 we describe the modeling assumptions involved in the passage from a time-dependent PDE to an neural quantum state-based solution of McLachlan's variational principle. The remainder of the paper is dedicated to numerical experiments, focusing on the problem of financial derivative pricing, which suffers from a curse-of-dimensionality. Section 4 in particular provides numerical confirmation in the case of correlated diffusions, and section 5 describes the application of these results to option pricing in the Black-Scholes pricing framework.

2. Theory

2.1. Generalities of McLachlan's variational principle

McLachlan's variational principle (McLachlan 1964) is an example of a time-dependent variational principle (TDVP) that approximates the solution of the TDSE by evolution within a space of parametrized trial functions. TDVPs for the TDSE have been devised for tensor network states (Haegeman *et al.* 2011), neural-network quantum states (NQS) (Carleo and Troyer 2017) and parametrized quantum circuits (Stokes *et al.* 2020, Barison *et al.* 2021). Ref. Stokes *et al.* (2022) studied TDVPs through the lens of information geometry providing a unified perspective applicable to variational quantum algorithms (VQAs) and variational quantum Monte Carlo with normalized neural-network quantum states. In the following section, we further generalize TDVPs to include general time-dependent PDEs and establish additional connections with the VQA and numerical analysis literature. Given a time-dependent PDE for the state variable $u(t,x) \in \mathbb{C}$ with $(t,x) \in [0,T] \times \Omega$, together with a choice of parametrized functions of the spatial variable $\{u_\theta : \Omega \longrightarrow \mathbb{C} \,|\, \theta \in \mathbb{R}^p\}$, the output of a TDVP is parametrized curve $\gamma : [0,T] \longrightarrow \mathbb{R}^p$ in the space of variational parameters such that $u_{\gamma(t)}$ optimally describes $u(t,\cdot)$ in some distance metric $\mathrm{dist}(\cdot,\cdot)$ for all $t \in [0,T]$. Consider the initial value problem for a general time-dependent PDE of the form,

$$\partial_t u(t,x) = \mathcal{F}(t,x,u) \tag{1}$$

$$u(0,x) = u_0 \in L^2(\Omega;\mathbb{C}), \tag{2}$$

with prescribed boundary conditions on $\partial\Omega$. In order to simplify exposition, in this section we avoid complications of boundary conditions by assuming either $\Omega = \mathbb{R}^d$ with suitable decay at infinity or $\Omega = \mathbb{T}^d$. In practice we replace the

† VQAs for mult-asset financial derivative pricing have been subsequently explored in Kubo *et al.* (2022).

spatial domain by a mesh $\widehat{\Omega} \subset \Omega$, thereby approximating square-integrable functions by square summable vectors. The imposition of boundary conditions on the spatial mesh is then achieved via the use of source functions. Let Φ_s^t denote the time evolution map for state variable such that $\Phi_u^t \circ \Phi_s^u = \Phi_s^t$ and in particular $u(t,\cdot) = \Phi_0^t(u_0)$. Given an initial parameter vector $\theta_0 \in \mathbb{R}^p$ and a step size $\delta t > 0$, define a sequence of parameter vectors $(\theta_k)_{k\in\mathbb{N}}$ by the following iteration

$$\theta_{k+1} := \underset{\theta\in\mathbb{R}^p}{arg\,min}\left[\text{dist}\left(\Phi_{k\delta t}^{(k+1)\delta t}(u_{\theta_k}), u_\theta\right)\right]. \tag{3}$$

In quantum physics applications, a suitable distance metric is the Fubini-Study metric, as previously argued both in VMC (Carleo and Troyer 2017, Stokes *et al.* 2022) and in VQA (Stokes *et al.* 2020, Barison *et al.* 2021) literature. In this work we choose dist$(\cdot,\cdot)$ to be the Euclidean norm. Rather than solving the discrete-time dynamical system (3) directly, we consider the limit of infinitesimal step size $\delta t \longrightarrow 0$ in which it reduces to the following system of ordinary differential equations (ODEs)† with initial condition $\gamma(0) = \theta_0$:

$$M(\gamma(t))\,\gamma'(t) = V(t,\gamma(t)) \tag{4}$$

where

$$\begin{aligned} M_{ij}(\theta) &:= \text{Re}\left[\left\langle \frac{\partial u_\theta}{\partial\theta_i}\middle|\frac{\partial u_\theta}{\partial\theta_j}\right\rangle\right], \\ V_i(t,\theta) &:= \text{Re}\left[\left\langle \frac{\partial u_\theta}{\partial\theta_i}\middle|\mathcal{F}(t,u_\theta)\right\rangle\right] \end{aligned} \tag{5}$$

and where $\langle\cdot|\cdot\rangle$, $\|\cdot\|_2$ denote the standard inner product and the induced norm for $L^2(\Omega,\mathbb{C})$, respectively. Although the matrix M_{ij} is necessarily positive semi-definite, it may be degenerate, reflecting the possibility of multiple minima in the optimization problem (3). Thus, regularization techniques are generally required in order to obtain a well-posed system of ODEs.

2.2. *Analogy with finite element approximations*

Before discussing our autoregressive neural-network quantum state implementation, it is useful to orient within scientific computing literature by showing that the formalism shares close parallels with the classic finite element method applied to the linear inhomogeneous case $\mathcal{F}(t,x,u) = \mathcal{L}u(t,x) + f(t,x)$. Given a set of real basis functions $\{\varphi_i\}_{i=1}^m$, define the variational family consisting of a weighted superposition,

$$u_\theta(x) = \sum_{i=1}^{m}\theta_i\varphi_i(x), \tag{6}$$

where $\theta \in \mathbb{R}^m$ is assumed here. Substituting the above into (4) one finds the following ODE determining the dynamics of the weights,

$$M\gamma'(t) = -K\gamma(t) + f(t), \quad M_{ij} := \langle\varphi_i|\varphi_j\rangle,$$

$$K_{ij} := -\langle\varphi_i|\mathcal{L}u_j\rangle, \quad f_i(t) := \langle\varphi_i|f(t)\rangle, \tag{7}$$

which can be recognized as the standard discrete linear systems arising in finite element methods, with M and K the mass- and stiffness-matrix, respectively. Recall that since finite element techniques use non-overlapping elements to discretize the spatial domain and employ basis functions with localized support, the problem size scales as $m \sim p^d$ in d spatial dimensions, where p is the average number of elements per dimension. Thus, the curse-of-dimensionality arises from need to perform increasingly high-dimensional, albeit sparse, linear algebra in order to solve the linear system (7). The approach advocated in the following section, in contrast, overcomes the curse-of-dimensionality by representing the solution vector in terms of an autoregressive neural-network quantum state.

2.3. *Neural-network quantum state implementation*

In order to overcome the curse of dimensionality with respect to the spatial dimension d, which is inherent in the utilization of a d-dimensional mesh $\widehat{\Omega} \subset \Omega \subseteq \mathbb{R}^d$, we utilize stochastic estimation combined with autoregressive assumptions. In particular, we take inspiration from both the variational quantum algorithm introduced in Alghassi *et al.* (2021) as well variational quantum Monte Carlo using autoregressive NQS (Hibat-Allah *et al.* 2020, Sharir *et al.* 2020), by parametrizing the solution of the PDE as $u_\theta = \alpha\,\psi_\beta$ where ψ_β is a unit-normalized NQS with variational parameters β and $\alpha > 0$ is a scale factor, whose time-dependence must be determined from the evolution equations along with β. The variational parameters thus consist of the augmented parameter vector $\theta := (\log\alpha, \beta) \in \mathbb{R}^{p+1}$ where $\beta \in \mathbb{R}^p$ is an unconstrained vector representing the weights and biases of the neural network. Plugging the rescaled ansatz into the evolution equations (4), one obtains an augmented system of first-order, non-linear ordinary differential equations determining the time-dependence of the augmented vector θ. The overlaps defining M and V are estimated using the VMC importance sampling technique, where the probability density is chosen to be the modulus-squared wavefunction $|\psi_\beta(x)|^2$.

3. Numerical implementation

In this section, we provide detailed modeling assumptions required to implement the algorithm. After describing a mesh-based encoding of the state variable into a multi-qubit state, we then introduce the model architecture and computation mechanisms including the forward pass and the autoregressive sampling process. Pre-training is necessary in order to satisfy the initial condition of the PDE, and we adopt the standard approach (Sirignano and Spiliopoulos 2018) by updating the model iteratively on batches of randomly selected mesh points. Finally, we describe the stochastic estimation procedure used to evolve the variational trial state using McLachlan's variational principle.

† See appendix 1 for a derivation. Similiar ODEs have been introduced in the neural Galerkin method (Bruna *et al.* 2022).

3.1. Conversion to meshed form

Rather than working in the continuum, we assume spatial discretization of the function $u(t,\cdot)$ on a regular grid $\widehat{\Omega}$ embedded in the d-dimensional domain $\Omega = [a_1, b_1] \times \cdots \times [a_d, b_d]$ with a total of 2^n grid points. Without loss of generality, we assume Ω is a regular hypercube satisfying $|b_1 - a_1| = \cdots = |b_d - a_d|$, and that the mesh size along each axis is $2^{n/d}$. In order to represent the state of the discretized field in terms of the state of an n-qubit system, we assign each computational basis state $|k_1, \ldots, k_n\rangle \in \mathbb{C}^{2^n}$ with $(k_1, \ldots, k_n) \in \{0,1\}^n$ to a linear index defined by $k = \sum_{i=1}^{n} k_i 2^i$ and then unravel the linear index to indices along each of d axes yielding a d-tuple $(\bar{k}_1, \ldots, \bar{k}_d) \in \{0, \ldots, n/d - 1\}^d$ defined by

$$\sum_{i=1}^{n} \bar{k}_i (2^{\frac{n}{d}})^{i-1} = k. \tag{8}$$

The mesh point $\boldsymbol{x}^k \in \widehat{\Omega}$ corresponding to the linear index $k \in \{0, 1, \ldots, 2^n - 1\}$ is thus

$$\boldsymbol{x}^k = (a_1 + \bar{k}_1 \Delta x, \ldots, a_d + \bar{k}_d \Delta x) \tag{9}$$

where $\Delta x = |b_1 - a_1|/2^{\frac{n}{d}}$. Henceforth, we do not distinguish the index k, the binary string $(k_1, \ldots, k_n)$ and the corresponding coordinate $\boldsymbol{x}^k \in \widehat{\Omega}$. The digitized representation of the state of the PDE at time t is thus the following unnormalized n-qubit state

$$|u(t)\rangle = \sum_{k \in \widehat{\Omega}} u(\boldsymbol{x}^k, t) |k_1, \ldots, k_n\rangle \tag{10}$$

and likewise the parametrized approximation $|u_\theta\rangle$ of $|u(t)\rangle$ is given by,

$$|u_\theta\rangle = \sum_{k \in \widehat{\Omega}} u_\theta(\boldsymbol{x}^k) |k_1, \ldots, k_n\rangle \tag{11}$$

where $u_\theta(\cdot)$ is a function defined on $\widehat{\Omega} \subset \Omega$.

3.2. Autoregressive assumption and sampling

In order to obtain an expressive family of trial functions $u_\theta : \widehat{\Omega} \longrightarrow \mathbb{C}$ which furthermore admits an efficient stochastic estimation procedure, we express u_θ as multiple of a unit-normalized neural-network quantum state $\psi_\beta : \widehat{\Omega} \longrightarrow \mathbb{C}$ with variational parameters $\beta \in \mathbb{R}^p$,

$$u_\theta(\boldsymbol{x}^k) = \alpha\, \psi_\beta(k_1, \ldots, k_n), \quad \|\psi_\beta\|_{\widehat{\Omega}} = 1. \tag{12}$$

A simple method to ensure unit-normalization and efficient sampling is to follow the work of MADE (Germain *et al.* 2015) which exploited a masked version of a fully connected layer, where some connections in the computational path are removed in order to satisfy the autoregressive properties, In particular, if we assume a choice of variables such that the state variable is strictly positive† $u_\theta(x) > 0$, then a suitable choice of unit-normalized function is

$$\psi_\beta(k_1, \ldots, k_n) = \prod_{i=1}^{n} \sqrt{p_{\beta,i}(k_i | k_{i-1}, \ldots, k_1)}, \tag{13}$$

where $p_{\beta,i}(\cdot | k_{i-1}, \ldots, k_1)$ is the parametrized conditional probability distribution for the ith bit.

3.3. Pre-training

Before we perform the time evolution, initial variational parameters θ_0 must be selected in order to match the variational function $u_{\theta_0}(x)$ with the choice of initial condition $u_0(x)$ for $x \in \widehat{\Omega}$ in the spatial domain. This can be achieved via optimization of the following objective function using stochastic gradient descent,

$$J(\alpha_0, \beta_0) = \big\| \alpha_0 |\psi_{\beta_0}\rangle - |u_0\rangle \big\|^2_{\widehat{\Omega}}. \tag{14}$$

3.4. Evolution

In this section we discuss the details of the stochastic estimation of M and V necessary to evolve the augmented parameter vector $\theta = (\log \alpha, \beta) \in \mathbb{R}^{p+1}$. For simplicity of presentation we consider the affine case, in which $\mathcal{F}$ separates into a linear partial differential operator $\mathcal{L}$ plus a source function f independent of the state variable

$$\mathcal{F}(t, x, u) = \mathcal{L}u(t, x) + f(t, x). \tag{15}$$

Consider the decomposition,

$$M = \left[\begin{array}{c|c} M_{00} & M_{0,1:p} \\ M_{1:p,0} & M_{1:p,1:p} \end{array}\right], \quad V = \left[\begin{array}{c} V_0 \\ V_{1:p} \end{array}\right]. \tag{16}$$

It is convenient to introduce the following helper functions. In particular, define the Born probability distribution $\rho_\beta(x) \in [0, \infty)$, the wavefunction score $\sigma_\beta(x) \in \mathbb{C}^n$ and the local energy $l_\theta(t, x) \in \mathbb{C}$ as follows,

$$\begin{aligned} \rho_\beta(x) &:= |\psi_\beta(x)|^2, \quad \sigma_\beta(x) := \frac{\nabla_\beta \psi_\beta(x)}{\psi_\beta(x)}, \\ l_\theta(t, x) &:= \frac{(\mathcal{L}\psi_\beta)(x)}{\psi_\beta(x)} + \frac{f(t, x)}{\alpha}. \end{aligned} \tag{17}$$

By straightforward calculus, we obtain

$$\begin{aligned} M_{00} &= \alpha^2, \quad M_{1:p,0} = M_{0,1:p} = \alpha^2 \operatorname{Re}\Big[\mathop{\mathbb{E}}_{x \sim \rho_\beta} \sigma_\beta(x) \Big], \\ M_{1:p,1:p} &= \alpha^2 \operatorname{Re}\Big[\mathop{\mathbb{E}}_{x \sim \rho_\beta} \overline{\sigma_\beta(x)}\, \sigma_\beta(x)^T \Big], \end{aligned} \tag{18}$$

† This can be considered as a special case of the complex-valued case (Hibat-Allah *et al.* 2020, Sharir *et al.* 2020).

and

$$V_0 = \alpha^2 \,\mathrm{Re}\left[\mathop{\mathbb{E}}_{x\sim\rho_\beta} l_\theta(t,x)\right],$$
$$V_{1:p} = \alpha^2 \,\mathrm{Re}\left[\mathop{\mathbb{E}}_{x\sim\rho_\beta} \overline{\sigma_\beta(x)}\, l_\theta(t,x)\right]. \tag{19}$$

The expectation values over x are approximated using Monte Carlo sampling. In practice, the batch of randomly generated samples is represented in the form of a buffer $\mathcal{B} = \{x_i\}_{i=1}^B$ that stores the unique samples in the batch and a counter $\mathcal{C} = \{c_i\}_{i=1}^B$ that records the number of occurrences of each of the corresponding samples. Expectation values are then approximated by sums of the following form,

$$\mathop{\mathbb{E}}_{x\sim\rho_\beta}[g(x)] \approx \frac{1}{\sum_{i=1}^B c_i} \sum_{i=1}^B c_i g(x_i). \tag{20}$$

Computation of per-sample gradient. In order to compute the dense representation of M efficiently, it is desirable to have access to the per-sample gradients $\{\nabla_\beta \psi_\beta(x_i)\}_{i=1}^B$, which is typically not directly accessible during a traditional backward pass by some deep learning software. In order to avoid inefficient forward and backward passes for each of the B samples, we utilize `BackPack` which collects the quantities necessary to compute the individual gradients and reuses them to compute the per-sample gradient without significant computational overhead.

Parallel extraction of matrix elements. Given a sample $x \in \widehat{\Omega}$, corresponding to a particular row-index of the operator $\widehat{\mathcal{L}} \in \mathbb{C}^{2^n\times 2^n}$, the calculation of the local energy $l_\theta(t,x)$ defined in (17) involves determining the nonzero entries of $\widehat{\mathcal{L}}$ in that row. Fortunately, the structure of the PDE problem ensures that the location and values of these entries can be determined in $O(\mathrm{poly}(n))$ time. We exploit CPU parallelization to determine the nonzero row entries for each sample in the batch. Note that the maximal number of nonzero entries per row is determined in advance; e.g. $2n^2+1$ for diffusion operator with Dirichlet boundary conditions.

Boundary Conditions. Boundary conditions are implemented on the grid using an appropriate choice of source function. In the case of Dirichlet conditions, for example, we choose the source function as follows

$$f(t,x) = u(t,x)\mathbb{1}_{\partial\Omega}(x) \tag{21}$$

where $\mathbb{1}_{\partial\Omega}$ denotes the indicator function for the set $\partial\Omega$. This choice of source function corresponds to imposing Dirichlet boundary conditions on the grid with specified boundary behavior $u(t,x)$ for $x \in \partial\Omega$.

Parameter update. The network can be trained by incrementing the parameters using a Euler scheme in the direction $\delta\theta \in \mathbb{R}^{p+1}$ given by the solution of the following linear system

$$M\delta\theta = V\delta t, \tag{22}$$

where M, V are computed as discussed before. In practice, M is usually ill-conditioned due to the fact that it's essentially a sum of B rank one matrices. To stabilize the inverse operation, we consider the singular value decomposition $M = U\Sigma W^T$, and remove the diagonal values of Σ smaller than a small threshold $\epsilon = 10^{-12}$ to obtain Σ_r of size $r\times r$, as well as U_r, W_r of size $(p+1)\times r$. The direction vector is therefore approximated by $W_r\Sigma_r^{-1}U_r^T V\,\delta t$.

4. Diffusion with Gaussian initializations

In this section, we show various numerical experiments demonstrating the convergence and run time analysis of our proposed approach. Here, we consider the d-dimensional heat equation for which $\mathcal{L} = D\nabla\cdot\nabla$ in (15) (diffusion constant set to $D = 0.1$) with either periodic or Dirichlet boundary conditions on $\partial\Omega$. For benchmarking purposes, we employ a finite difference method with the standard central difference scheme to discretize $\mathcal{L}$ (formulas given in Appendix 2) and forward Euler for time-stepping.

4.1. Experimental setup

For initialization we chose a discrete isotropic Gaussian, expressed in terms of the modified Bessel function $I_x(t)$ of integer order x,

$$u_0(x) = \prod_{i=1}^d e^{-t} I_{x_i}(t), \tag{23}$$

where $t > 0$ is a parameter controlling the width of the Gaussian, and x ranges from 0 to 2^{n-1}. Pre-training was performed using Adam optimizer (Kingma and Ba 2015) for 50k iterations with batch size 128, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate was warm-started for the first 1/10 total training iterations, then decayed by a factor of 10 at 3/7, 5/7 of the total training iterations.

After completion of pre-training, the state was evolved for a total evolution time $T = 1$ using a step size of $\delta t = 5\times 10^{-5}$ and a batch size of $B = 1024$ for Monte Carlo estimation on each iteration. The time development of the state was compared to the result of Euler time-stepping the initial condition (23) using the same step-size. Due to the exponential scaling of the matrix $\widehat{\mathcal{L}}$, we only establish this baseline for the number of qubits $n \le 16$. All reported numerical results are averaged across exactly 5 seeds, each trained for 20k steps.

Table 1 reports the relative error between our method and forward Euler method over 20k iterations, for diffusion problem with Dirichlet boundary conditions. The relative error of the obtained solution is computed by comparing the norm of the space-time history of the variational state with the result of Euler development,

$$\text{error} := \frac{1}{T}\sum_{t\in\widehat{T}} \frac{\big\||u(t)\rangle - |u_{\gamma(t)}\rangle\big\|_{\widehat{\Omega}}}{\big\||u(t)\rangle\big\|_{\widehat{\Omega}}}. \tag{24}$$

In general, the overall solution obtained using our method matches well with that obtained using forward Euler method. Their discrepancy increases as the dimensionality of the problem increases; nonetheless, within a tolerable threshold (e.g. less than 10%). One remedy is to simply increase the batch size, for example, we show in Section 4.4 that the relative error improves from 15% to around 3% with a batch size that is ten times larger.

4.2. Running time analysis

Since we discretize the domain Ω using a grid of size $|\widehat{\Omega}| = 2^n$, the time complexity of forward Euler method, expressed in terms of n is $O(T \times 2^{2n})$, where T is the number of iterations. The proposed VMC algorithm, in contrast, scales as $O(TB\,\mathrm{poly}(n))$, where B is the batch size. In more detail, the forward pass scales as $O(TBn^2)$, and the sampling is $O(TBn^3)$, due to the sequential nature of the auto-regressive sampling process. This polynomial scaling comes at a price of the approximate nature of the time evolution step, and the implicit access to entries of the state vector compared to the forward Euler method which offers $O(1)$ lookup to entries of the state.

In table 2, we report the average running time of both forward Euler method and our method for 2k iterations (one-tenth of the total running time). The batch size B of our method is fixed to be 500. Note that we cannot apply Euler for higher dimensions due to the memory constraint. Although forward Euler method is effective for small-scale problems, its complexity suffers from the exponential growth with respect to dimension d. The computational cost of our method originates from four sources: sampling, forward pass, per-sample gradient computation (backward pass), and extraction of matrix element information. All these sources contribute to the overhead time that causes our method to run slower in comparison with respect to forward Euler method for problems in lower dimensions. However, as the dimensionality increases, the run time of our method grows only at a polynomial rate. It can be seen from table 2 that our method is already faster than forward Euler for problem sizes characterized by $n = 16$ qubits.

4.3. Convergence visualization

We provide snapshots of our method for 2D diffusion problems with periodic boundary conditions over 20k iterations. We run our algorithm with five distinct initializations and record the snapshot every 2k iterations. In particular we do not employ any regularization techniques to enforce that the solution satisfies the boundary condition. It can be observed in figure 1 that our method successfully obeys the periodic boundary conditions.

4.4. Ablation study on batch size

Recall that M and V are approximated with Monte Carlo sampling using batches of unique samples. Intuitively, a larger batch size yields a better approximation to the exact expectation value, thereby providing more accurate model updates. In this section, we study the effect of batch size on the performance of our method. In the LHS of figure 2, the running time of our algorithm increases for both larger problem sizes and batch sizes. Note that the actual running time does not grow linearly with respect to the batch size in the plot due to the cache and parallelization. In the RHS of figure 2, we report the average relative error between the forward Euler method and the VMC method with various batch sizes. Given a fixed problem size (e.g. 4 dimensions with 4 qubits per dimension), we observe a performance improvement by increasing the batch size, which verifies our hypothesis that increasing the batch size does effectively improve the performance. Given a fixed batch size, our method performs worse as the dimensionality of the problem increases. This result implies that we need a larger batch size to guarantee good performance for problems in higher dimensions.

5. Option pricing

In this section, we explain how the numerical results obtained in Section 4 can be applied to the multi-dimensional Black-Scholes model. In Section 5.1 we present the Black-Scholes stochastic model and provide the Black-Scholes PDE for a general European option in this model. We reduce the

Table 1. Average relative error of our method in comparison with forward Euler method over 20k iterations.

Operator	Boundary Condition	n/d	# of Dimensions d			
			1	2	3	4
Diffusion	Dirichlet	4	5.13×10^{-3}	7.92×10^{-3}	3.12×10^{-2}	1.47×10^{-1}
Diffusion	Dirichlet	5	2.91×10^{-3}	9.91×10^{-3}	7.24×10^{-2}	–

Note: The proposed algorithm solution is compared with a Euler forward method for the diffusion equation over 20k iterations. For forward Euler method, the time step is 5×10^{-5} with a total time of 1. The relative error is computed as $\frac{1}{T}\sum_{t=1}^{T} \frac{\|u(t,x)-f(x;\theta_t)\|}{\|u(t,x)\|}$.

Table 2. Average running time over 2 k iterations.

Operator	Method	n/d	# of Dimensions d								
			1	2	3	4	5	6	7	8	9
Diffusion	Euler	4	0.024	0.137	1.559	305.92	–	–	–	–	–
Diffusion	Euler	5	0.031	0.225	68.827	–	–	–	–	–	–
Diffusion	Ours	4	29.52	55.76	138.88	230.02	360.80	518.00	792.63	1214.09	1812.94
Diffusion	Ours	5	32.53	88.75	173.43	311.82	507.93	847.85	1355.16	2379.86	4123.61

Notes: The batch size used here is 500. Forward Euler method suffers from the exponential complexity, whereas our method, despite having an overhead running time, enjoys a polynomial scaling. Note that we cannot apply Euler for higher dimensions due to the memory constraint.

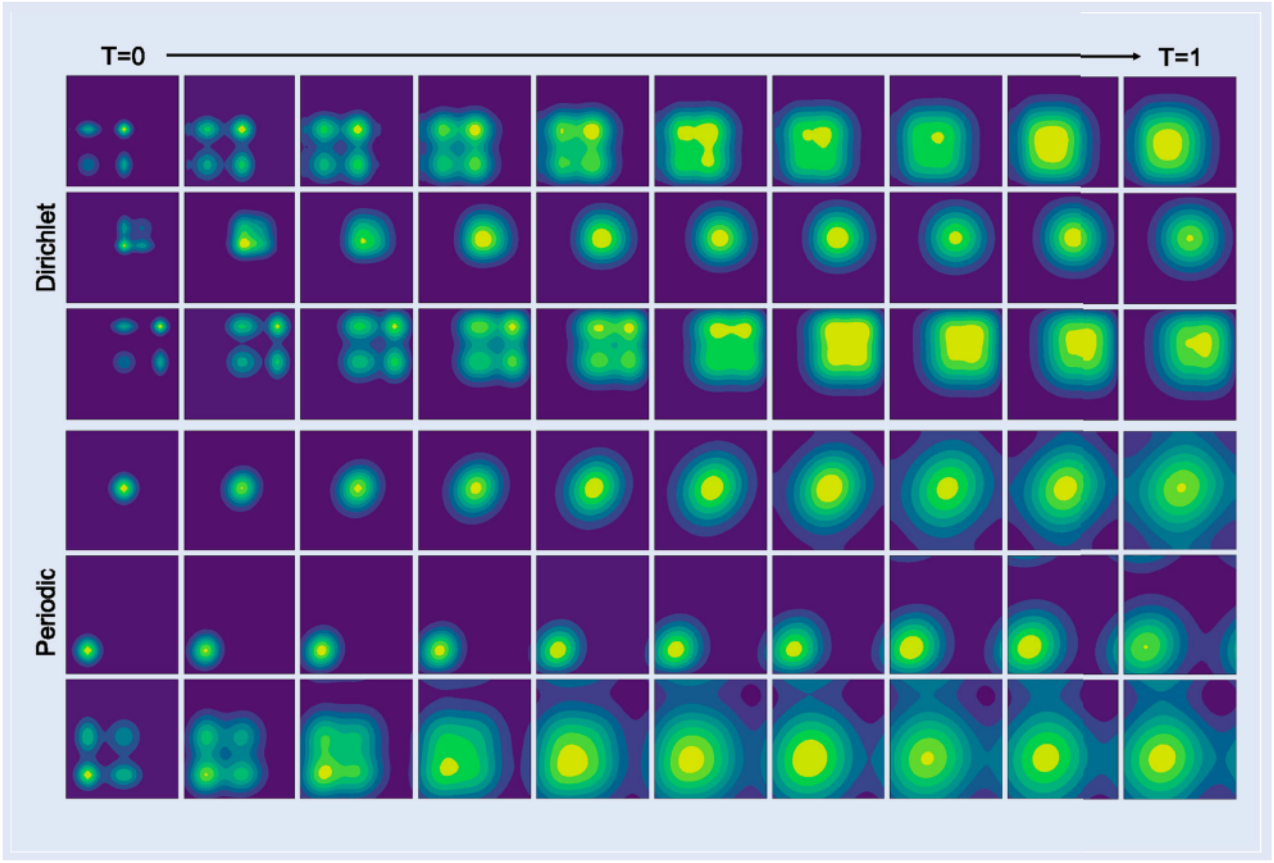


Figure 1. Snapshots of the evolution obtained using our algorithm for diffusion equation with Dirichlet and periodic boundary conditions and different choices of initialization.

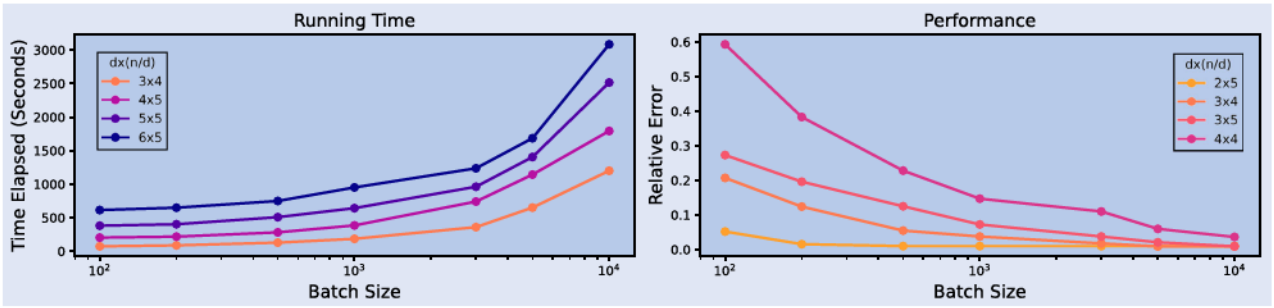


Figure 2. We report the running time and the average relative error with the forward Euler method over various batch sizes, for qubit sizes $n = d \times \frac{n}{d}$, where d is the dimensionality of the problem. The running time grows with respect to the batch size and the problem size. On the other hand, the performance is greatly improved when training with larger batch sizes.

multi-dimensional PDE to the standard heat equation. Then, in Section 5.2, we apply our algorithm to the heat equation and translate it back to the Black-Scholes equation.

5.1. Option pricing in Black-Scholes model

For this we consider the Black-Scholes model that consists of a risk-free asset with a constant risk-free return $r > 0$ and d risky assets whose dynamics are given by

$$\mathrm{d}S_t^i = \mu_i S_t^i \mathrm{d}t + \sigma_i S_t^i \,\mathrm{d}W_t^i, \quad i = 1, \dots, d.$$

The parameters μ_i and σ_i, $i = 1, \dots, d$, are constants, and $\{W^i\}_{i=1}^d$ are Wiener processes with quadratic covariation $[W_t^i, W_t^j] = \rho_{ij} t$. Consider further a European option, whose payment at (the predetermined) expiry time $T > 0$ is $\Psi(S_T^1, \dots, S_T^d)$, for some measurable function Ψ. Let V be the conditional price of this option, i.e. $V(t, x_1, \dots, x_d)$ is the price for the option at time t, given that $S_t^i = x_i$ for $i = 1, \dots, d$. It is well-known that V satisfies the following Black-Scholes PDE:

$$\begin{cases} \dfrac{\partial V}{\partial t} + \displaystyle\sum_{i=1}^{d} r x_i \frac{\partial V}{\partial x_i} + \frac{1}{2} \sum_{i=1}^{d} \sigma_i^2 x_i^2 \frac{\partial^2 V}{\partial x_i^2} \\ \quad + \displaystyle\sum_{i \neq j} \frac{1}{2} \rho_{ij} \sigma_i \sigma_j x_i x_j \frac{\partial^2 V}{\partial x_i \partial x_j} - rV = 0, \qquad (t, x) \in [0, T] \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \times (0, \infty)^d, \\ V(T, x_1, \dots, x_d) = \Psi(x_1, \dots, x_d). \end{cases}$$

Following Guillaume (2019), we may reduce this n-dimensional equation to the n-dimensional standard heat

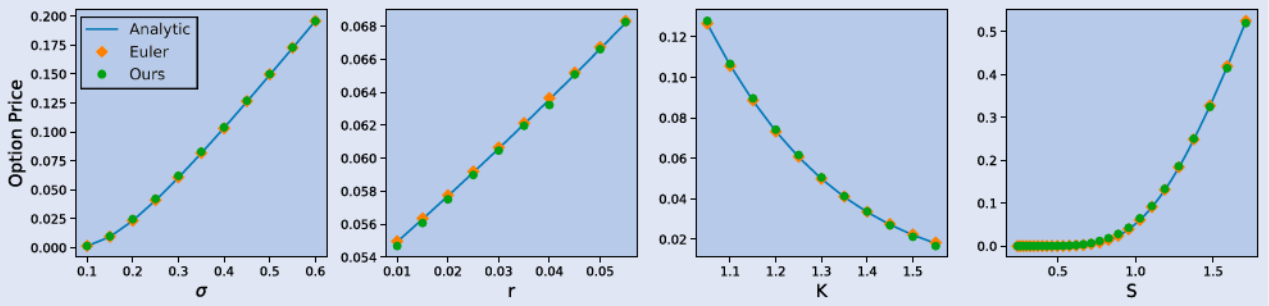


Figure 3. Ablation study on volatility σ, interest rate r, strike price K and initial price. We fix a base setting with hyper-parameters $\sigma = 0.3, r = 0.03, K = 1.25$, and run our algorithm on each setting with only one hyper-parameter deviated from the base setting. In addition, we plot the wave function under the base setting. We compare our solution at the execution time T versus forward Euler method and the corresponding analytic ground truth. Our method is robust under all settings and achieves satisfactory performance.

equation. To this end, set

$$u(t, y_1, \dots, y_n) = V(T - t, e^{\sigma_1 y_1}, \dots, e^{\sigma_n y_n}) e^{\sum_{i=1}^{n} -a_i \sigma_i y_i - bt},$$

where a_i and b satisfy the following system of equations:

$$\begin{aligned} &\sum_{i=1}^{d} a_i \left(r - \frac{\sigma_i^2}{2} \right) + \frac{1}{2} \sum_{i=1}^{d} a_i^2 \sigma_i^2 \\ &\quad + \sum_{i \neq j} \frac{1}{2} \rho_{ij} \sigma_i \sigma_j a_i a_j - r - b = 0, \\ &r - \frac{\sigma_i^2}{2} + a_i \sigma_i^2 + \sum_{j \neq i} \rho_{ij} \sigma_i \sigma_j a_j = 0, \quad i = 1, \dots, d. \end{aligned}$$

Then, u satisfies the following heat equation

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{1}{2} \displaystyle\sum_{i=1}^{d} \frac{\partial^2 u}{\partial y_i^2} + \sum_{i \neq j} \frac{1}{2} \rho_{ij} \frac{\partial^2 u}{\partial y_i \partial y_j}, & (t, y) \in [0, T] \times \mathbb{R}^d, \\ u(0) = \Psi(e^{\sigma_1 y_1}, \dots, e^{\sigma_d y_d}) e^{-\sum a_i \sigma_i y_i}. \end{cases}$$

with $u(0) = \Psi(e^{\sigma_1 y_1}, \dots, e^{\sigma_d y_d}) e^{-\sum a_i \sigma_i y_i}$.

5.2. *Numerical experiments*

In this section, we apply our algorithm to option pricing in Black-Scholes model across different settings using the numerical solution to the heat equation and the translation from the heat equation to the Black-Scholes equation from the previous subsection. We test the performance of our algorithm as well as show a calculation of the option price for higher dimensions.

In figure 3 and table 3 we provide examples to test the performance of our algorithm. In both we include 1D examples for the price of a European call option, whose value at the expiry time T is $V(T, s) = \Psi(s) = \max\{s - K, 0\}$, where K is a predetermined constant, called the strike price. We vary the volatility σ, strike price K, interest rate r, expiry time T, and initial price of the stock S. In figure 3, we compare against the forward Euler method and the ground truth *Black-Scholes formula*, which admits an analytical solution in 1D. Specifically,

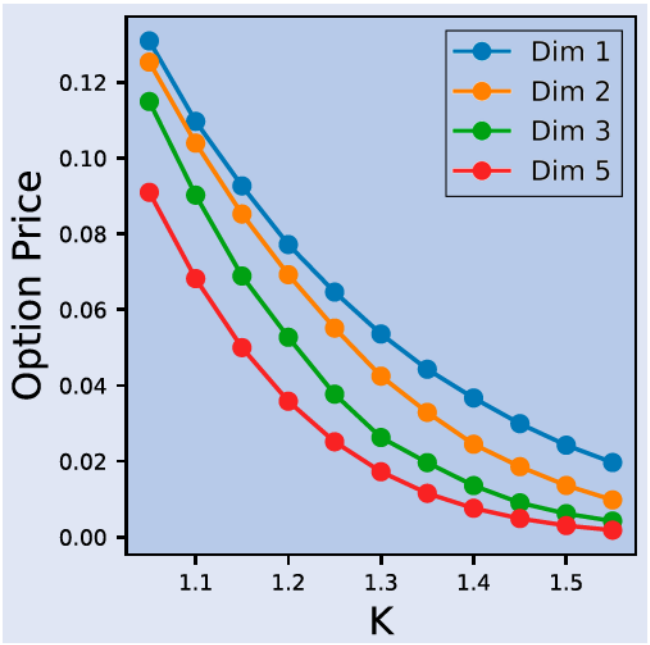


Figure 4. Ablation study on dimensionality, following the base settings in figure 3, $\sigma = 0.3, r = 0.03, K = 1.25$.

it is given by

$$V(t, s) = N(d_+)s - N(d_-)Ke^{-r(T-t)}, \tag{25}$$

where N is the cumulative distribution function of the standard normal distribution and

$$d_\pm = \frac{\ln \frac{s}{K} + (r \pm \frac{\sigma^2}{2})(T - t)}{\sigma \sqrt{T - t}}.$$

Our method is robust under all settings and achieves satisfactory performance. In table 3 we compare our accuracy against the forward Euler method, where we use the same analytical solution. Although the performance of our algorithm is inferior to that of Euler, it still achieves good accuracy and the margin can be treated as a price to pay for reducing the exponential scaling down to a polynomial one. Our approach is valuable for high dimensions, where other methods, such as the forward Euler method, suffer from the curse of dimensionality. Table 3 also includes 2D examples with four different options: *Basket European call* and *put*, *rainbow max European call*, and *spread European put*, whose payoffs are listed

Table 3. List of experiments for the application of our algorithm to Black–Scholes equation.

Problem	Initial & Boundary Cond.	d	D	T	r	K	σ	Ours	Euler
Black–Scholes 1D	1D CALL	1	–	1	0.03	1.25	0.3	0.011781	0.002494
Black–Scholes 1D	1D CALL	1	–	1	0.03	1.25	0.1	0.032792	0.000930
Black–Scholes 1D	1D CALL	1	–	1	0.03	1.25	0.2	0.017272	0.002389
Black–Scholes 1D	1D CALL	1	–	1	0.03	1.25	0.4	0.011560	0.001392
Black–Scholes 1D	1D CALL	1	–	1	0.03	1.05	0.3	0.086764	0.001750
Black–Scholes 1D	1D CALL	1	–	1	0.03	1.15	0.3	0.013365	0.002521
Black–Scholes 1D	1D CALL	1	–	1	0.03	1.35	0.3	0.012260	0.000328
Black–Scholes 1D	1D CALL	1	–	1	0.03	1.45	0.3	0.012626	0.000845
Black–Scholes 1D	1D CALL	1	–	1	0.01	1.25	0.3	0.010436	0.002538
Black–Scholes 1D	1D CALL	1	–	1	0.02	1.25	0.3	0.010599	0.002515
Black–Scholes 1D	1D CALL	1	–	1	0.04	1.25	0.3	0.014192	0.002477
Black–Scholes 1D	1D CALL	1	–	1	0.05	1.25	0.3	0.017423	0.002463
Black–Scholes 1D	1D CALL	1	–	0.5	0.03	1.25	0.3	0.022481	0.018385
Black–Scholes 1D	1D CALL	1	–	1.5	0.03	1.25	0.3	0.012049	0.014881
Black–Scholes 2D	2D BASKET CALL	2	0.1	1	0.03	1.25	0.3	0.053477	–
Black–Scholes 2D	2D BASKET PUT	2	0.1	1	0.03	1.25	0.3	0.043926	–
Black–Scholes 2D	2D RAINBOW MAX CALL	2	0.1	1	0.03	1.25	0.3	0.057949	–
Black–Scholes 2D	2D SPREAD PUT	2	0.1	1	0.03	1.25	0.3	0.031574	–

Notes: The hyper-parameters for the experiments are listed. We compute the relative error of our method (**Ours**) at expiration time T with analytical ground truth in the 1D case and Euler solution in the 2D case, respectively. For 1D, we also report the relative error of the forward Euler method with respect to the analytical ground truth (**Euler**) for comparison. Our algorithm achieves robust performance across various settings.

Table 4. Payoff functions for our experiments.

Option Type	Payoff Function at expiry $\Psi(s)$
1D Call	$\max(s-K,0)$
Basket Call	$\max(\sum w_i s_i - K, 0)$
Basket Put	$\max(K - \sum w_i s_i, 0)$
Rainbow Max Call	$\max(\max s_i - K, 0)$
2D Spread Put	$\max(K-(s_1-s_2),0)$

Notes: We consider basket call and put, Rainbow max call, and spread put options.

in table 4. Note that as we don't have an analytical solution for this case, the relative errors with respect to Euler solutions are reported instead. In figure 4, we provide a graph for the price of a basket European call option with up to five underlying stocks as a function of the strike price K. As expected the prices are convex with K.

Note that the Black-Scholes PDE lives on the positive orthant of $\mathbb{R}^d$ while its translation to the heat equation lives on $\mathbb{R}^d$. Thereby, all numerical algorithms, including ours, require artificial truncation of the domain. We choose the hypercube domain to be $[s_l, s_u]^d = [Ke^{-3\sigma_i\sqrt{T}}, Ke^{3\sigma_i\sqrt{T}}]^d$. This choice implies that s_l is small (close to $0+$) and s_u is large (close $+\infty$). On the faces of the hypercube, we use the time-discounted payoff functions, as they are reasonably accurate approximations of boundary values of the options considered. Given the number of qubits n and the hypercube input domain for the heat equation $[L_l, L_u]^d$, which is approximately $[-5,5]^d$, the mesh size of each axis is $(L_u - L_l)/(2^{n/d}+1)$.

6. Conclusions

In summary, we introduced a generalization of McLachlan's variational principle applicable to generic time-dependent PDEs as well as a quantum-inspired training algorithm based on neural-network quantum states which can be used to perform approximate time evolution in high dimensions, overcoming the curse-of-dimensionality. Although we focused on a mesh-based formulation in which the quantum state vector is represented by n qubits, it is clear that the mesh is not mandated by the formulation and it would be very interesting to pursue meshless variants based on continuous-variable neural-network quantum states including normalizing flows (Stokes *et al.* 2022) and to address non-trivial boundary conditions. There exist a number of directions in which the results of this paper can be potentially improved. Since we only considered a first-order Euler approximation of the ODE (4) it would be natural to incorporate high-order time stepping schemes (e.g.Runge-Kutte methods). As an alternative, it would be interesting to pursue a direct solution of the discrete-time dynamical system (3) which has proven successful in both the VMC (Gutiérrez and Mendl 2022) and VQA (Barison *et al.* 2021) literature.

Acknowledgments

The Authors thank the anonymous AE and the referees for their suggestions, which helped to improve our paper.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

Authors gratefully acknowledge support from NSF under grants DMS-2038030 and DMS-2006305. This research was

supported in part through computational resources and services provided by the Advanced Research Computing (ARC) at the University of Michigan.

ORCID

Asaf Cohen http://orcid.org/0000-0002-9211-7956
Shravan Veerapaneni http://orcid.org/0000-0002-2294-7233

References

Alghassi, H., Deshmukh, A., Ibrahim, N., Robles, N., Woerner, S. and Zoufal, C., A variational quantum algorithm for the Feynman-Kac formula. arXiv preprint arXiv:2108.10846, 2021.

Barison, S., Vicentini, F. and Carleo, G., An efficient quantum algorithm for the time evolution of parameterized circuits. *Quantum*, 2021, **5**, 512. http://dx.doi.org/10.22331/q-2021-07-28-512

Beck, C., Weinan, E and Jentzen, A., Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *J. Nonlinear Sci.*, 2019, **29**(4), 1563–1619. http://dx.doi.org/10.1007/s00332-018-9525-3

Bruna, J., Peherstorfer, B. and Vanden-Eijnden, E., Neural Galerkin scheme with active learning for high-dimensional evolution equations. arXiv preprint arXiv:2203.01360, 2022.

Carleo, G. and Troyer, M., Solving the quantum many-body problem with artificial neural networks. *Science*, 2017, **355**(6325), 602–606.

Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S.C., Endo, S., Fujii, K., McClean, J.R., Mitarai, K., Yuan, X., Cincio, L. and Coles, P.J., Variational quantum algorithms. *Nat. Rev. Phys.*, 2021, **3**(9), 625–644.

Fontanela, F., Jacquier, A. and Oumgari, M., A quantum algorithm for linear PDEs arising in finance. *SIAM J. Financ. Math.*, 2021, **12**(4), SC98–SC114.

Germain, M., Gregor, K., Murray, I. and Larochelle, H., MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889, 2015 (PMLR).

Guillaume, T., On the multidimensional Black-Scholes partial differential equation. *Ann. Oper. Res.*, 2019, **281**(1–2), 229–251. http://dx.doi.org/10.1007/s10479-018-3001-1

Gutiérrez, I.L. and Mendl, C.B., Real time evolution with neural-network quantum states. *Quantum*, 2022, **6**, 627.

Haegeman, J., Cirac, J.I., Osborne, T.J., Pižorn, I., Verschelde, H. and Verstraete, F., Time-dependent variational principle for quantum lattices. *Phys. Rev. Lett.*, 2011, **107**(7), 070601.

Han, J., Jentzen, A. and Weinan, E., Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci. USA*, 2018, **115**(34), 8505–8510. http://dx.doi.org/10.1073/pnas.1718942115

Harrow, A.W., Hassidim, A. and Lloyd, S., Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 2009, **103**(15), 150502.

Hibat-Allah, M., Ganahl, M., Hayward, L.E., Melko, R.G. and Carrasquilla, J., Recurrent neural network wave functions. *Phys. Rev. Res.*, 2020, **2**(2), 023358.

Huré, C., Pham, H. and Warin, X., Deep backward schemes for high-dimensional nonlinear PDEs. *Math. Comp.*, 2020, **89**(324), 1547–1579. http://dx.doi.org/10.1090/mcom/3514

Kingma, D.P. and Ba, J., Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations*, 2015 (San Diego, CA, USA).

Kubo, K., Miyamoto, K., Mitarai, K. and Fujii, K., Pricing multi-asset derivatives by variational quantum algorithms. arXiv preprint arXiv:2207.01277, 2022.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. and Anandkumar, A., Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895, 2020.

McLachlan, A., A variational solution of the time-dependent Schrodinger equation. *Mol. Phys.*, 1964, **8**(1), 39–44.

Preskill, J., Quantum computing in the nisq era and beyond. *Quantum*, 2018, **2**, 79.

Raissi, M., Perdikaris, P. and Karniadakis, G.E., Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 2019, **378**, 686–707.

Reh, M. and Gärttner, M., Variational monte carlo approach to partial differential equations with neural networks. arXiv preprint arXiv:2206.01927, 2022.

Sharir, O., Levine, Y., Wies, N., Carleo, G. and Shashua, A., Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Phys. Rev. Lett.*, 2020, **124**(2), 020503.

Sirignano, J. and Spiliopoulos, K., Dgm: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.*, 2018, **375**, 1339–1364.

Stokes, J., Chen, B. and Veerapaneni, S., Numerical and geometrical aspects of flow-based variational quantum monte carlo. arXiv preprint arXiv:2203.14824, 2022.

Stokes, J., Izaac, J., Killoran, N. and Carleo, G., Quantum natural gradient. *Quantum*, 2020, **4**, 269.

Wang, J., Chen, Z., Luo, D., Zhao, Z., Hur, V.M. and Clark, B.K., Spacetime neural network for high dimensional quantum dynamics. arXiv preprint arXiv:2108.02200, 2021.

Weinan, E., Han, J. and Jentzen, A., Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.*, 2017, **5**(4), 349–380. http://dx.doi.org/10.1007/s40304-017-0117-6

Appendices

Appendix 1. Derivation of evolution equations

Assume that the time evolution map and the variational trial function admit Taylor expansions of the form

$$\Phi_t^{t+\delta t}(u) = u + \mathcal{F}(t,u)\delta t + \mathcal{O}(\delta t^2), \tag{A1}$$

$$u_{\theta+\delta\theta} = u_\theta + \sum_{i=1}^{p} \frac{\partial u_\theta}{\partial \theta_i}\delta\theta_i + \mathcal{O}(\delta\theta^2). \tag{A2}$$

Then

$$\begin{aligned}
&\left\| \Phi_t^{t+\delta t}(u_\theta) - u_{\theta+\delta\theta} \right\|_2^2 \\
&= \sum_{i,j=1}^{p} \left\langle \frac{\partial u_\theta}{\partial \theta_i} \middle| \frac{\partial u_\theta}{\partial \theta_j} \right\rangle \delta\theta_i\delta\theta_j - \sum_{i=1}^{p} \left[\left\langle \mathcal{F}(t,u_\theta) \middle| \frac{\partial u_\theta}{\partial \theta_i} \right\rangle \right. \\
&\quad \left. + \left\langle \frac{\partial u_\theta}{\partial \theta_i} \middle| \mathcal{F}(t,u_\theta) \right\rangle \right] \delta t\, \delta\theta_i + \cdots \\
&= \sum_{i,j=1}^{p} \frac{1}{2} \left[\left\langle \frac{\partial u_\theta}{\partial \theta_i} \middle| \frac{\partial u_\theta}{\partial \theta_j} \right\rangle + \left\langle \frac{\partial u_\theta}{\partial \theta_j} \middle| \frac{\partial u_\theta}{\partial \theta_i} \right\rangle \right] \delta\theta_i\delta\theta_j \\
&\quad - \sum_{i=1}^{p} \left[\left\langle \mathcal{F}(t,u_\theta) \middle| \frac{\partial u_\theta}{\partial \theta_i} \right\rangle + \left\langle \frac{\partial u_\theta}{\partial \theta_i} \middle| \mathcal{F}(t,u_\theta) \right\rangle \right] \delta t\, \delta\theta_i + \cdots \\
&= \sum_{i,j=1}^{p} M_{ij}(\theta)\delta\theta_i\delta\theta_j - 2\delta t \sum_{i=1}^{p} \delta\theta_i V_i(t,\theta) + \cdots
\end{aligned} \tag{A3}$$

where in the last line we used the conjugate-symmetry of $\langle\cdot|\cdot\rangle$ and we have neglected $\delta\theta$-independent terms and terms higher than

quadratic order in the multi-variable Taylor expansion in $\delta\theta$ and δt. The first-order optimality condition $0 = \frac{\partial}{\partial \delta\theta_i} \|\Phi_t^{t+\delta t}(u_\theta) - u_{\theta+\delta\theta}\|_2^2$, gives, at lowest order in $\delta\theta$ and δt,

$$0 = 2\sum_{j=1}^{p} M_{ij}(\theta)\delta\theta_j - 2V_i(t,\theta)\delta t + \cdots \tag{A4}$$

and thus taking the limit $\delta t \longrightarrow 0$ gives the result.

Appendix 2. Matrix representation of $\mathcal{L}$

In d spatial dimensions and multi-index $\mathbf{i} = \{i_1, i_2, \ldots, i_d\}$, let $\mathbf{i} \pm \mathbf{e}_k = \{i_1, i_2, \ldots, i_k \pm 1, \ldots, i_d\}$ and $\mathbf{i} \pm \mathbf{e}_k \pm \mathbf{e}_{k'} = \{i_1, i_2, \ldots, i_k \pm 1, \ldots, i_{k'} \pm 1, \ldots, i_d\}$. Notice we do not allow $+1$ if $i_k = \frac{n}{d}$ or -1 if $i_k = 1$. Then the elements of the matrix is given by:

$$[\widehat{\mathcal{L}}]_{\mathbf{i},\mathbf{j}} = \begin{cases} \frac{-d}{\Delta^2}, & \mathbf{j} = \mathbf{i}, \\ \frac{1}{2\Delta^2}, & \mathbf{j} = \mathbf{i} \pm \mathbf{e}_k, \\ \frac{D}{4\Delta^2}, & \mathbf{j} = \mathbf{i} \pm \mathbf{e}_k \pm \mathbf{e}_{k'}, \\ \frac{-D}{4\Delta^2}, & \mathbf{j} = \mathbf{i} \pm \mathbf{e}_k \mp \mathbf{e}_{k'}, \\ 0, & \text{otherwise.} \end{cases} \tag{A5}$$

Appendix 3. Additional numerical details

The network architecture was chosen to be MADE (Germain *et al.* 2015), taking the one-dimensional state as input and outputing the logarithmic probability amplitudes $\{\log p_{\beta,i}(k_i|k_{i-1}, \ldots, k_1)\}_{i=1}^n$ as described in (13). Note that both input and output are vectors of the same size. The structure of MADE is as follows

$$\textit{Input} \xrightarrow{[B,n]} \texttt{MaskedFC}_{n,h} \xrightarrow{[B,h]} \texttt{ReLU}$$
$$\xrightarrow{[B,h]} \texttt{MaskedFC}_{h,n} \xrightarrow{[B,n]} \texttt{LogSigmoid} \xrightarrow{[B,n]} \textit{Output},$$

Here B is the batch size and n is the number of qubits. $\texttt{MaskedFC}_{a,b}$ is a masked fully connected layer with input size a and output size b, removing the connections in the computational path of MADE. We chose the latent size h to scale with the input size as $h = 2n$. The depth of our model is 2 throughout all experiments in this paper.

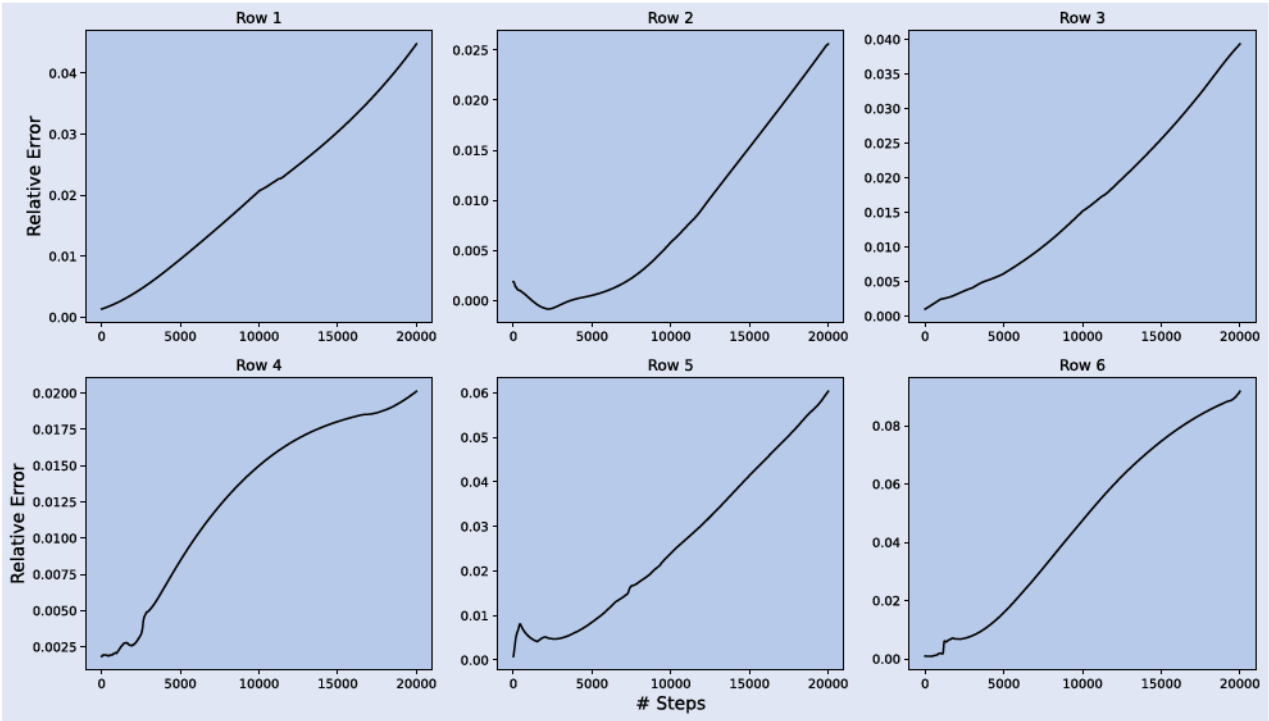


Figure A1. The relative error in the VMC approximate solution as a function of step size compared to the Euler method for each experiments shown in figure 1.