MACHINE LEARNING
Science and Technology

PURPOSE-LED PUBLISHING™

Part of the MACHINE LEARNING Series

PAPER • OPEN ACCESS

Self-adaptive physics-informed quantum machine learning for solving differential equations

To cite this article: Abhishek Setty *et al* 2025 *Mach. Learn.: Sci. Technol.* **6** 015002

View the article online for updates and enhancements.

You may also like

- QCPINN: quantum-classical physics-informed neural networks for solving PDEs
Afrah Farea, Saiful Khan and Mustafa Serdar Celebi

- Quantum Fourier networks for solving parametric PDEs
Nishant Jain, Jonas Landman, Natansh Mathur et al.

- Design of a novel quantum neural network
Xu-Feng Niu and Wen-Ping Ma

This content was downloaded from IP address 134.94.123.71 on 29/06/2026 at 17:19

IOP Publishing Mach. Learn.: Sci. Technol. 6 (2025) 015002 https://doi.org/10.1088/2632-2153/ada3ab

MACHINE LEARNING Science and Technology

PAPER

Self-adaptive physics-informed quantum machine learning for solving differential equations

Abhishek Setty[1,2,3,*], Rasul Abdusalamov[1] and Felix Motzoi[2,3]

[1] Department of Continuum Mechanics, RWTH Aachen University, Aachen 52062, Germany
[2] Forschungszentrum Jülich, Institute of Quantum Control (PGI-8), Jülich D-52425, Germany
[3] Institute for Theoretical Physics, University of Cologne, Cologne D-50937, Germany
[*] Author to whom any correspondence should be addressed.

E-mail: a.setty@fz-juelich.de

Keywords: quantum machine learning, physics-informed neural networks, variational quantum algorithms, differential equations

OPEN ACCESS

RECEIVED
22 October 2024

REVISED
28 November 2024

ACCEPTED FOR PUBLICATION
27 December 2024

PUBLISHED
13 January 2025

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Abstract

Chebyshev polynomials have shown significant promise as an efficient tool for both classical and quantum neural networks to solve linear and nonlinear differential equations (DEs). In this work, we adapt and generalize this framework in a quantum machine learning setting for a variety of problems, including the 2D Poisson's equation, second-order linear DE, system of DEs, nonlinear Duffing and Riccati equation. In particular, we propose in the quantum setting a modified Self-Adaptive Physics-Informed Neural Network approach, where self-adaptive weights are applied to problems with multi-objective loss functions. We further explore capturing correlations in our loss function using a quantum-correlated measurement, resulting in improved accuracy for initial value problems. We analyse also the use of entangling layers and their impact on the solution accuracy for second-order DEs. The results indicate a promising approach to the near-term evaluation of DEs on quantum devices.

1. Introduction

With the invention of calculus, a new era of mathematics was created by introducing differential equations (DEs) by Sir Isaac Newton and Gottfried Leibniz [1, 2]. DEs are ubiquitous in describing physical phenomena, beginning with classical mechanics, fluid dynamics, electromagnetism, and quantum mechanics. To date, the search for efficient and accurate solutions to DEs has remained a challenging task and many problems have no closed-form solution. Numerical methods have been introduced in the last century, such as the finite element method, finite differences, or finite volume method that can be used to solve more complex systems [3–6]. These methods remain crucial for many engineering applications and are further investigated. Nevertheless, they also have limitations, such as the problem with discretization errors, convergence rates, and the requirement for large computational resources.

With the recent rise in machine learning (ML) algorithms, a novel concept was introduced by Raissi *et al* where physical principles were combined with data-driven techniques to solve partial DEs (PDE) [7]. The key idea is to implement the DE together with the physical constraints of the system as additional loss terms during the neural network training. The introduced neural networks are known as physics-informed neural networks (PINNs) and have been used for a variety of applications such as for solving inverse problems involving nonlinear PDEs [8], problems in fluid mechanics [9] or for the analysis of defects in materials [10]. PINNs can handle noisy or incomplete data and do not rely on any fine-grained numerical discretization. Although PINNs have a promising future, one major problem remains: training the networks is related to immense computation costs.

© 2025 The Author(s). Published by IOP Publishing Ltd

Meanwhile, recent breakthroughs in quantum computing have suggested computational advantages in ML and solving linear systems of equations. The seminal Harrow–Hassidim–Lloyd algorithm (HHL) [11, 12] provides a system solution that is encoded into the amplitude of a quantum state through quantum phase estimation and amplitude amplification. This method has been extended to solve nonlinear DEs in computational fluid dynamics and structural mechanics [13, 14]. An alternative method for solving DEs also using amplitude encoding employs instead employing variational quantum circuits [15, 16], where a quantum nonlinear programming unit may efficiently calculate nonlinear product operators of the field, even when these nonlinearities are large. A drawback of this method is however that error can propagate iteratively in the current step depending on the previous step solution while having to deal also with relatively deep (though narrow) circuits. Lastly, quantum kernel methods have been proposed to solve regression as well as DEs [17], leading to an advantageous choice for the loss function that allows for efficient multi-point training. Additional advances have been made in resolving PDEs in the context of quantum amplitude estimation, employing Chebyshev points [18]. The resolution of advection–diffusion equations through the use of variational quantum linear systems has been introduced in [19].

Recently, a different approach has been proposed that avoids problems from amplitude encoding by using feature map encoding to approximate functions and solve nonlinear first-order DEs using quantum circuits [20–22]. The approach encodes the state using a latent space representation defined in the high-dimensional Hilbert space, benefiting from the large expressive power of quantum feature maps and parameterized quantum circuits.

In our work, we extend the usage of the quantum Chebyshev feature map while solving a variety of DEs. In addition, we incorporate the approach of self-adaptive weights for balancing multi-objective loss functions while solving DEs. Particularly, we modify the method of self-adaptive weights introduced for classical PINNs [23] into the setting of the proposed variational quantum algorithm. Moreover, we study the use of correlated measurement observables and more entangling layers for better convergence. Our results show that these approaches lead to a measurable improvement in the convergence and accuracy for solving the given test cases, namely the Duffing equation, Riccati equation, system of DEs, second-order linear DE, and the 2D Poisson's equation.

2. Methods

The variational quantum circuit to solve DEs proposed by Kyriienko *et al* [21] is presented in figure 1. In this circuit, there are three main parts, namely, a quantum feature map circuit for encoding Chebyshev polynomials, a variational circuit optimizing the cost function, and measurement (where we test correlated observables in the quantum loss functions). An overview of the variables used in this work is mentioned in table 1 for the sake of clarity.

For a single independent variable x_i, the quantum state for this circuit will be $|f_{\phi,\theta}(x_i)\rangle = \hat{\mathcal{U}}_\theta \hat{\mathcal{U}}_\phi(x_i)|0\rangle$. The real-valued classical function $f(x_i)$ is calculated by expectation value of a predefined Hermitian cost operator $\hat{\mathcal{C}}$, such that, $f(x_i) = \langle f_{\phi,\theta}(x_i)|\hat{\mathcal{C}}|f_{\phi,\theta}(x_i)\rangle$. Solving DEs requires computing gradients of desired functions with respect to the independent variable, $\frac{\mathrm{d}f(x)}{\mathrm{d}x}$. For the given circuit, this derivative requires considering quantum feature map ($\phi(x)$), such as, $\frac{\mathrm{d}\hat{\mathcal{U}}_\phi(x)}{\mathrm{d}x} = \sum_j \hat{\mathcal{U}}_{\mathrm{d}\phi,j}(x)$. The Chebyshev feature map which encodes Chebyshev polynomials into the circuit is given by

$$\hat{\mathcal{U}}_\phi(x) = \bigotimes_{j=1}^{N} R_{Y,j}(2j \arccos x), \tag{1}$$

where j is the qubit index. In-depth details of this approach are available in appendix B.

2.1. Derivation of second order derivative

For solving second-order DEs, we need to compute second-order derivatives of circuits. For a problem solved in section 3.3, the second derivative $\frac{\mathrm{d}^2 u_{\mathrm{pred}}}{\mathrm{d}(x_f^i)^2}$ in equation (24) considering the quantum Chebyshev feature map (refer equation (1)) is calculated using chain rule as follows,

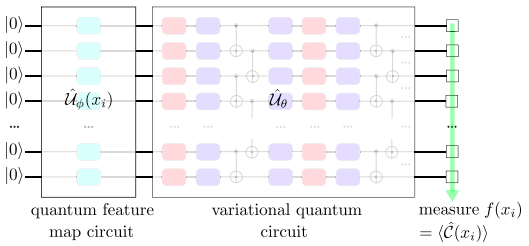

Figure 1. Quantum circuit for solving DEs. On the left, a feature map ($\phi(x_i)$) block is used for encoding the value of a function at a specific input x_i, followed by a variational parameter block $\hat{\mathcal{U}}_\theta$, in which θ is a set of variational parameters optimized over cost function. In the end, the desired function ($f(x_i)$) in the DE is evaluated as an expectation value of the circuit based on a chosen operator $\hat{\mathcal{C}}$.

Table 1. Overview of used variables.

Variables	
N_f	Set of collocation points
N_b	Set of boundary points
x_f^i	Collocation grid points
x_b^i	Boundary points
u/u_{true}	True solution
u_{pred}	Predicted solution
α_f	Residual loss weight
α_b	Boundary loss weight
$\hat{\mathcal{U}}_\phi$	Quantum feature map
$\hat{\mathcal{U}}_\theta$	Variational quantum circuit

$$\begin{aligned}\frac{\mathrm{d}^2 u_{\text{pred}}}{\mathrm{d}\left(x_f^i\right)^2} &= \frac{\mathrm{d}}{\mathrm{d}x_f^i}\left(\frac{\mathrm{d}u_{\text{pred}}}{\mathrm{d}x_f^i}\right),\\ &= \frac{\mathrm{d}}{\mathrm{d}x_f^i}\left(\sum_j \frac{\mathrm{d}u_{\text{pred}}}{\mathrm{d}\phi_j\left(x_f^i\right)}\frac{\mathrm{d}\phi_j\left(x_f^i\right)}{\mathrm{d}x_f^i}\right),\\ &= \sum_{j,k}\frac{\mathrm{d}^2 u_{\text{pred}}}{\mathrm{d}\phi_j\left(x_f^i\right)\mathrm{d}\phi_k\left(x_f^i\right)}\left[\frac{4jk}{1-\left(x_f^i\right)^2}\right]\\ &\quad+\sum_j \frac{\mathrm{d}u_{\text{pred}}}{\mathrm{d}\phi_j\left(x_f^i\right)}\left[\frac{-2jx_f^i}{\left(1-\left(x_f^i\right)^2\right)^{\frac{3}{2}}}\right].\end{aligned} \tag{2}$$

Notice that equation (2) contains a term where the denominator can become zero in case of $x_f^i = 1.0$ leading to numerical instability. Therefore in the problem of solving second-order DE (refer section 3.3), the last collocation point is chosen with $x_f^{30} = 0.99$. Correspondingly, the Dirichlet boundary conditions are evaluated at this point.

2.2. Self-adaptive weights for multi-objective loss function

In the weighted multi-objective loss function, the weights balancing the total loss function can be treated as variables instead of constants as shown in appendix A. The Self-Adaptive PINNs [23] also referred to as SAPINNs, allow these weights to be trainable and they are applied to each training point individually. The idea behind SAPINNs is to maximize the weights and minimize the losses. We explore the applicability of this classical method for the variational quantum algorithm proposed in this work. If the variational parameters of the quantum circuit are given by $\boldsymbol{\theta}$, then the total loss function can be expressed by

$$\mathcal{L}\left(\boldsymbol{\theta},\boldsymbol{\lambda}_f,\boldsymbol{\lambda}_b\right) = \mathcal{L}_f\left(\boldsymbol{\theta},\boldsymbol{\lambda}_f\right) + \mathcal{L}_b\left(\boldsymbol{\theta},\boldsymbol{\lambda}_b\right), \tag{3}$$

where $\boldsymbol{\lambda}_f = (\lambda_f^1, \cdots, \lambda_f^{N_f})$ and $\boldsymbol{\lambda}_b = (\lambda_b^1, \cdots, \lambda_b^{N_b})$ are trainable, non-negative self-adaptation weights for the residual, and boundary points, respectively. These loss terms are given by

$$\mathcal{L}_f(\boldsymbol{\theta}, \boldsymbol{\lambda}_f) = \frac{1}{N_f} \sum_{i=1}^{N_f} m\left(\lambda_f^i\right) \left(f\left(x_f^i; \boldsymbol{\theta}\right)\right)^2, \tag{4}$$

$$\mathcal{L}_b(\boldsymbol{\theta}, \boldsymbol{\lambda}_b) = \frac{1}{N_b} \sum_{i=1}^{N_b} m\left(\lambda_b^i\right) \left(u_{\text{pred}}\left(x_b^i; \boldsymbol{\theta}\right) - u_b\left(x_b^i\right)\right)^2, \tag{5}$$

where the self-adaptation mask function $m(\lambda)$ defined on $[0, \infty)$ is a non-negative, differentiable, strictly increasing function of λ. An interesting point to note about SAPINNs is that the loss $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}_f, \boldsymbol{\lambda}_b)$ is minimized with respect to the variational parameters $\boldsymbol{\theta}$, as usual, but is maximized with respect to the self-adaptation weights $\boldsymbol{\lambda}_f, \boldsymbol{\lambda}_b$. All parameters are updated by a gradient descent approach with

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta^k \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}^k, \boldsymbol{\lambda}_f^k, \boldsymbol{\lambda}_b^k\right), \tag{6}$$

$$\boldsymbol{\lambda}_f^{k+1} = \boldsymbol{\lambda}_f^k + \rho_f^k \nabla_{\boldsymbol{\lambda}_f} \mathcal{L}\left(\boldsymbol{\theta}^k, \boldsymbol{\lambda}_f^k, \boldsymbol{\lambda}_b^k\right), \tag{7}$$

$$\boldsymbol{\lambda}_b^{k+1} = \boldsymbol{\lambda}_b^k + \rho_b^k \nabla_{\boldsymbol{\lambda}_b} \mathcal{L}\left(\boldsymbol{\theta}^k, \boldsymbol{\lambda}_f^k, \boldsymbol{\lambda}_b^k\right), \tag{8}$$

where $\eta^k > 0$ is the learning rate for the variational parameters at step k, $\rho_p^k = 0.01$ is a separate learning rate chosen for the self-adaptation weights for $p = f, b$. Furthermore, the gradients are described as

$$\nabla_{\boldsymbol{\lambda}_f} \mathcal{L}\left(\boldsymbol{\theta}^k, \boldsymbol{\lambda}_f^k, \boldsymbol{\lambda}_b^k\right) = \frac{1}{N_f} \begin{bmatrix} m'\left(\lambda_f^{k,1}\right) \left(f\left(x_f^1; \boldsymbol{\theta}\right)\right)^2 \\ \cdots \\ m'\left(\lambda_f^{k,N_f}\right) \left(f\left(x_f^{N_f}; \boldsymbol{\theta}\right)\right)^2 \end{bmatrix}, \tag{9}$$

$$\nabla_{\boldsymbol{\lambda}_b} \mathcal{L}\left(\boldsymbol{w}^k, \boldsymbol{\lambda}_f^k, \boldsymbol{\lambda}_b^k\right) = \frac{1}{N_b} \begin{bmatrix} m'\left(\lambda_b^{k,1}\right) \left(u_{\text{pred}}\left(x_b^1; \boldsymbol{\theta}\right) - u_b\left(x_b^1\right)\right)^2 \\ \cdots \\ m'\left(\lambda_b^{k,N_b}\right) \left(u_{\text{pred}}\left(x_b^{N_b}; \boldsymbol{\theta}\right) - u_b\left(x_b^{N_b}\right)\right)^2 \end{bmatrix}. \tag{10}$$

2.3. Mask functions for self-adaptive weights

The self-adaptive weights method works on employing mask functions for each loss term in the multi-objective loss function. These mask functions act as adaptive weights that penalize each loss term that is not converging during optimization. A mask function consists of variational parameters λ which are optimized along with variational quantum circuit parameters.

For the problem solved in section 3.3, two masked functions m have been tested: a polynomial mask (referred to as *Polynomial*) and a logistic mask (referred to as *Logistic*). Since the polynomial mask is subjected to numerical overflow, these masks are modified with threshold values for λ, over which the curve flattens (see figure 6(a)). The hyperparameters of polynomial mask $m(\lambda_i) = c\lambda_i^q$ for the residual and boundary conditions are heuristically chosen. All λ are initialized to 1 and the threshold values $\lambda_{\max}$ are chosen accordingly. For the polynomial mask functions, the values for residual are chosen as $c_r = 0.1$, $q_r = 2$, and $\lambda_{\max,r} = 10$. For the boundary conditions these parameters are provided with $c_b = 1000$, $q_b = 2$, and $\lambda_{\max,b} = 1000$. In the case of the logistic mask functions, the saturation value for the residual is chosen as 0.1, and for boundary conditions, it is provided with 1000.

2.4. Solving PDE

Solving a PDE requires encoding multiple independent variables into the quantum feature map. Therefore, a novel quantum circuit is needed, as shown in figure 2. That is, the desired function $u(x, y)$ requires quantum feature mapping of two independent variables x and y. Therefore, we define two unitary operators $\hat{\mathcal{U}}_{\phi_x}(x)$ and $\hat{\mathcal{U}}_{\phi_y}(y)$ for each nonlinear feature map $\phi_x(x)$ and $\phi_y(y)$. The combined unitary operator is written as, $\hat{\mathcal{U}}_{\phi_{x,y}}(x, y) = \hat{\mathcal{U}}_{\phi_x}(x) \otimes \hat{\mathcal{U}}_{\phi_y}(y)$. The state of the quantum circuit for these two variables is given by

$$|f_{\phi_{x,y},\theta}(x, y)\rangle = \hat{\mathcal{U}}_\theta \hat{\mathcal{U}}_{\phi_{x,y}}(x, y)|0\rangle. \tag{11}$$

Therefore, the expectation value of the circuit with the Hamiltonian $\hat{\mathcal{C}}$ is given as,

$$f(x, y) = \langle f_{\phi_{x,y},\theta}(x, y)|\hat{\mathcal{C}}|f_{\phi_{x,y},\theta}(x, y)\rangle. \tag{12}$$

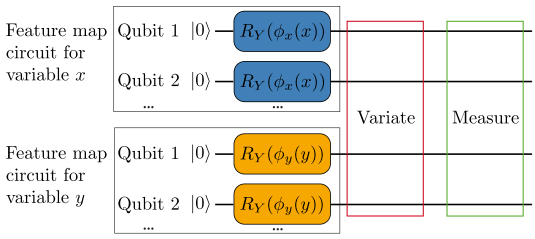

Figure 2. Visualization of a quantum circuit for encoding two independent variables using separate feature maps $\hat{\mathcal{U}}_{\phi_x}(x) = \bigotimes_{j=1}^{N_x} R_{Y,j}(\phi_x(x))$ and $\hat{\mathcal{U}}_{\phi_y}(y) = \bigotimes_{m=1}^{N_y} R_{Y,m}(\phi_y(y))$. The qubit numbering for each feature map starts from 1 as they must be encoded as independent Chebyshev polynomials into the circuit. Also, the variational part of the quantum circuit (in red box) and measurement of the circuit (in green box) are represented following the feature map blocks.

The feature maps in this study, $\phi_x(x)$ and $\phi_y(y)$ are both Chebyshev feature maps, $\phi_{xj}(x) = 2j\arccos(x)$ and $\phi_{ym}(y) = 2m\arccos(y)$, respectively. The unitary operator encoding the feature map for variable x is described as $\hat{\mathcal{U}}_{\phi_x}(x) = \bigotimes_{j=1}^{N_x} R_{Y,j}(2j\arccos x)$ (i.e. as the top box in figure 2). Similarly, for variable y is described as $\hat{\mathcal{U}}_{\phi_y}(x) = \bigotimes_{m=1}^{N_y} R_{Y,m}(2m\arccos x)$ (i.e. as the bottom box in figure 2). Here N_x and N_y are the number of qubits for each independent variable. The combined observable is the summation of Pauli-Z operators of all qubits such as $\hat{C} = \sum_j Z_j + \sum_m Z_m$.

For the problem solved in section 3.5, we considered a 2D Poisson's equation equation (27). The Laplace term in Poisson's equation equation (27) considering the quantum Chebyshev feature maps (see equation (1)) is given by,

$$\begin{aligned}\Delta u(x,y) = &\sum_{j,k} \frac{\partial^2 u_{\text{pred}}}{\partial\phi_{xj}\left(x_f^i\right)\partial\phi_{xk}\left(x_f^i\right)}\left[\frac{4jk}{1-\left(x_f^i\right)^2}\right] \\ &+\sum_{j} \frac{\partial u_{\text{pred}}}{\partial\phi_{xj}\left(x_f^i\right)}\left[\frac{-2jx_f^i}{\left(1-\left(x_f^i\right)^2\right)^{\frac{3}{2}}}\right] \\ &+\sum_{m,n} \frac{\partial^2 u_{\text{pred}}}{\partial\phi_{ym}\left(y_f^i\right)\partial\phi_{yn}\left(y_f^i\right)}\left[\frac{4mn}{1-\left(y_f^i\right)^2}\right] \\ &+\sum_{m} \frac{\partial u_{\text{pred}}}{\partial\phi_{ym}\left(y_f^i\right)}\left[\frac{-2my_f^i}{\left(1-\left(y_f^i\right)^2\right)^{\frac{3}{2}}}\right],\end{aligned} \tag{13}$$

where, $j,k = 1,2,3,\ldots$ and $m,n = 1,2,3,\ldots$ are qubit indices for each feature map. Index $i = 1,2,\ldots,N_f$ corresponds to collocation points.

The calculation of the first and second-order derivatives of quantum circuits concerning all qubits are described by equations (14) and (15),

$$\begin{bmatrix}\frac{\partial u}{\partial x}\\ \frac{\partial u}{\partial y}\end{bmatrix} = \begin{bmatrix}\frac{\partial u_{\text{pred}}}{\partial\phi_{xj}\left(x_f^i\right)}\\ \frac{\partial u_{\text{pred}}}{\partial\phi_{ym}\left(y_f^i\right)}\end{bmatrix}, \tag{14}$$

$$\begin{bmatrix}\frac{\partial^2 u}{\partial x^2} & \frac{\partial^2 u}{\partial x\partial y}\\ \frac{\partial^2 u}{\partial y\partial x} & \frac{\partial^2 u}{\partial y^2}\end{bmatrix} = \begin{bmatrix}\frac{\partial^2 u_{\text{pred}}}{\partial\phi_{xj}\left(x_f^i\right)\partial\phi_{xk}\left(x_f^i\right)} & \frac{\partial^2 u_{\text{pred}}}{\partial\phi_{xj}\left(x_f^i\right)\partial\phi_{yn}\left(y_f^i\right)}\\ \frac{\partial^2 u_{\text{pred}}}{\partial\phi_{ym}\left(y_f^i\right)\partial\phi_{xk}\left(x_f^i\right)} & \frac{\partial^2 u_{\text{pred}}}{\partial\phi_{ym}\left(y_f^i\right)\partial\phi_{yn}\left(y_f^i\right)}\end{bmatrix}. \tag{15}$$

3. Results

For this work, we investigate and benchmark our proposed methodology on selected problems, based on earlier literature [21, 24]. We start with a highly nontrivial dynamics problem such as the Riccati equation

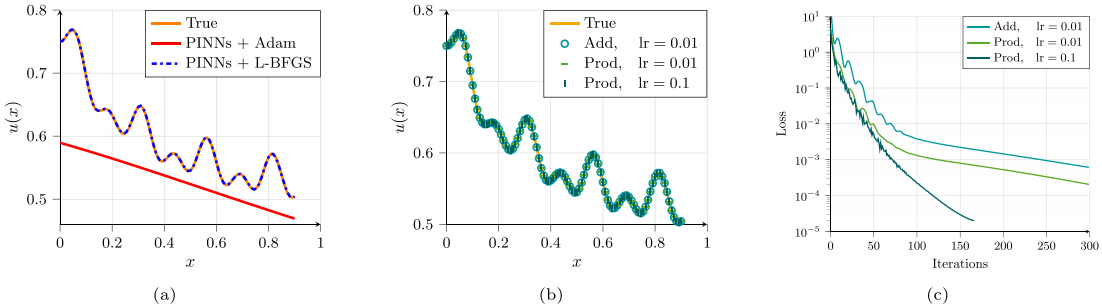

Figure 3. Solution of Riccati equation equation (16) and optimizer convergence. (a) Visualization of comparison between true solution obtained numerically by the Runge–Kutta method and a classical machine learning approach using PINNs with Adam and L-BFGS optimizer, (b) results for the quantum approach using the summation $\hat{\mathcal{C}} = \sum_j^N Z_j$ (referred as Add) and tensor product $\hat{\mathcal{C}} = \bigotimes_{j=1}^N Z_j$ (referred as Prod) of Pauli-Z operators for different learning rates (lr) and (c) their corresponding losses over the number of optimization iterations.

which is quadratic in an unknown function. Then we consider a coupled first-order DE, formulated as a system of DEs. In addition, a second-order linear DE and the nonlinear Duffing equation are investigated. Finally, a problem on the two-dimensional Poisson's equation presented by Kharazmi *et al* [24] is analyzed. All simulations are performed using Pennylane software [25].

3.1. Complex dynamics of Riccati equation

As a first example, a nonlinear DE of the general form known as the Riccati equation is considered, which has a rapidly oscillating non-periodic solution. This DE is given for a function $u(x)$ depending on the variable x as an initial value problem with

$$\frac{\mathrm{d}u}{\mathrm{d}x} - 4u + 6u^2 - \sin(50x) - u\cos(25x) + \frac{1}{2} = 0, \qquad u(x=0) = u_0 \tag{16}$$

where the initial condition is given by $u_0 = 0.75$. The domain $x \in [0, 0.9]$ is discretized into 100 equidistant points.

This DE is first solved by two classical methods: first, numerically by the explicit Runge–Kutta method of third order [26], which is considered the true solution for further comparisons, and second, using PINNs, which is considered as the classical ML solution. In the case of PINNs, a two-layer feed-forward neural network with 32 hidden neurons was considered, which has a tanh activation function in each layer. To solve this problem using PINNs, the loss from initial conditions and DEs is simply added as a multi-objective loss function appendix A. The results are visualized in figure 3(a). We first carried out the optimization with the Adam optimizer and found that it was unable to converge to the desired solution. However, when we carried out the optimization with the L-BFGS algorithm, the desired solution was reached.

For the quantum solution, a quantum circuit with 8 qubits and a depth of 8 blocks in the variational circuit with a circular entanglement layout (refer to appendix B.2) has been considered. The DE has terms proportional to u and u^2, which enables incorporating the initial condition into floating boundary handling (refer to appendix B.5). This enables faster convergence since a single combined loss function is used instead of a slower multi-objective optimization [21].

A set of $N_f = 100$ collocation points with $x_f^i \in \{1, \cdots, N_f\}$ are generated by equidistant grid points. The initial condition is given at x_f^1. Consider the circuit output as u_{pred}, and we define the modified output u_{m} such that

$$u_{\text{m}}\left(x_f^i\right) = \left(u_0 - u_{\text{pred}}\left(x_f^1\right)\right) + u_{\text{pred}}\left(x_f^i\right), \tag{17}$$

and the residual is given by

$$f\left(x_f^i\right) = \frac{\mathrm{d}u_{\text{pred}}}{\mathrm{d}x_f^i} - 4u_{\text{m}}\left(x_f^i\right) + 6\left(u_{\text{m}}\left(x_f^i\right)\right)^2 - \sin\left(50x_f^i\right) - u_{\text{m}}\cos\left(25x_f^i\right) + 1/2 \tag{18}$$

which we try to minimize. The results in figure 3(b) indicate that the quantum circuit can match the numerical Runge–Kutta solution as well as the solution determined classically using PINNs with L-BFGS optimizer. For all three approaches, the desired solution was obtained. The optimizer used in the quantum ML case was Adam, which was able to successfully converge to the desired solution unlike in the classical ML case where the very high precision update direction given by L-BFGS had to be used instead.

To further improve the performance of the quantum ansatz, we investigate the potential advantage of using a more expressive measurement basis to capture correlations in the circuit outcomes. In the case of using the local observable such as addition of Pauli-Z operators, the expectation values of each Pauli-Z operator are measured and summed up to obtain the overall result. If the quantum state is entangled, the summation of each Pauli-Z expectation value will not be able to capture the correlations between qubits. On the contrary, by employing the global observable such as choosing the Hamiltonian as the tensor product of Pauli-Z operators, we can capture the quantum correlations in the outcomes through entanglement in the quantum state. This modification can lead to better convergence of the cost function and improved accuracy of the solution.

In figure 3(c), we compare the convergence rate towards the true solution for the same initial variational parameters of the quantum circuit, the effect of two different observables, and variation in the learning rate. We see that maintaining the same learning rate as before, the standard sum basis of Pauli-Z operators $\hat{\mathcal{C}} = \sum_j Z_j$ is outperformed by measuring instead in a product Pauli-Z basis $\hat{\mathcal{C}} = \bigotimes_{j=1}^{N} Z_j$, with errors about half an order of magnitude better. The basis further allows for the use of a much larger learning rate than before, and increasing from a rate of 0.01 to 0.1, we observe dramatically faster convergence.

Thus, the Riccati equation is a promising first example, where the quantum approach can effectively encode the solutions, especially in systems where fast multi-partite encoding can be obtained before the measurement.

3.2. System of DEs

We now turn our methodology to another problem, first proposed in Kyriienko *et al* [21], where a first-order system of DEs is investigated. In this case, a vector $\boldsymbol{u}(x) = \left[u_1(x), u_2(x)\right]$ describes two modes, whose progression in time is x given by

$$F_1\left(\mathrm{d}_x\boldsymbol{u}, \boldsymbol{u}, x\right) = \frac{\mathrm{d}u_1}{\mathrm{d}x} - a_1 u_2 - a_2 u_1 = 0\,, \tag{19}$$

$$F_2\left[\mathrm{d}_x\boldsymbol{u}, \boldsymbol{u}, x\right] = \frac{\mathrm{d}u_2}{\mathrm{d}x} + a_2 u_2 + a_1 u_1 = 0\,, \tag{20}$$

$$\boldsymbol{u}(0) = \left[u_1(0), u_2(0)\right] = \left[u_{1,0}, u_{2,0}\right]\,, \tag{21}$$

where a_1 and a_2 are coupling parameters and $u_{1,0}$ as well as $u_{2,0}$ are initial conditions. The classical solution for this problem is obtained through ODE solvers by [27, 28]. To achieve a quantum solution, a quantum circuit comprising three qubits and a depth of seven blocks in the variational circuit is considered (see appendix B.2). An Adam optimizer is once again chosen, with a learning rate of 0.1. The observables are compared for the cases of addition and product as explained in the previous example. The domain is considered to be $x \in [0, 0.9]$, discretized into 100 equidistant grid points. The parameters for this problem are $a_1 = 5$, $a_2 = 3$ and initial conditions are $u_{1,0} = 0.5$ and $u_{2,0} = 0$. The results obtained from global observable are shown in figure 4(a). It is clear that the proposed quantum algorithm accurately captures the known solution. In figure 4(b), we again compare the modified observables such as the tensor product with the summation of Pauli-Z operators. Once again we see a large impact in the accuracy and convergence towards the known solution.

3.3. Second-order linear DE

As a further generalization, we now consider a second-order DE given for the function $u(x)$ with

$$\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} + 4\pi^2 \sin\left(2\pi x\right) = 0, \tag{22}$$

where $u(0) = 0$ and $u(0.99) = \sin 2\pi\, 0.99$. The analytical solution is given with $u(x) = \sin\left(2\pi x\right)$. Note that the Dirichlet boundary conditions at the right end are evaluated at $x = 0.99$ instead of $x = 1.0$ due to the condition explained in section 2.1. A set of $N_f = 30$ collocation points with $x_f^i \in \{1, \ldots, N_f\}$ are generated by equidistant grid points. Using the Chebyshev feature map

$$\phi_j\left(x_f^i\right) = 2j \arccos\left(x_f^i\right), \tag{23}$$

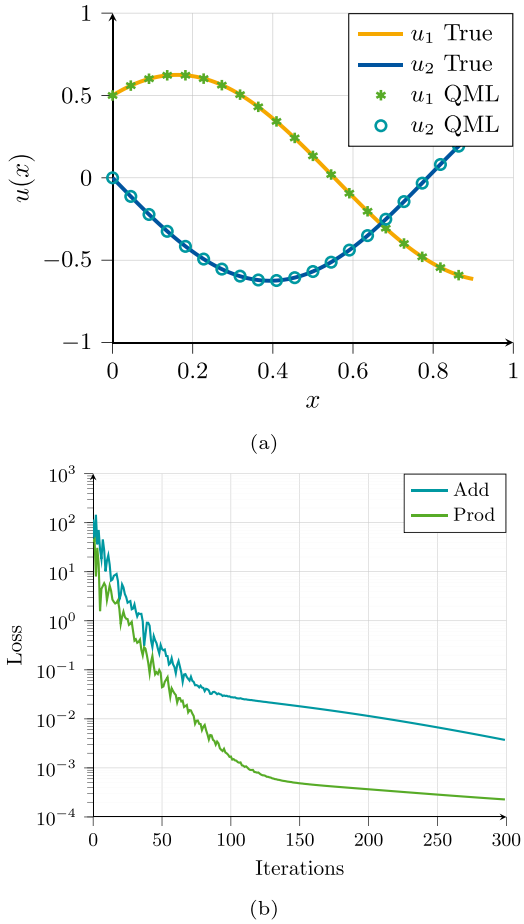

Figure 4. Solving of a system of coupled differential equations (refer equations (19)–(21)). Visualization of (a) comparison between classical and quantum solution for $u_1(x)$ and $u_2(x)$ as well as (b) the loss for the summation and tensor product of Pauli-Z operators cases over the number of optimization iterations.

the residual $f(x_f^i)$ for a single data point is given for the circuit output u_{pred} by

$$f\left(x_f^i\right) = \frac{\mathrm{d}^2 u_{\text{pred}}}{\mathrm{d}\left(x_f^i\right)^2} + 4\pi^2 \sin\left(2\pi x_f^i\right), \tag{24}$$

where the second derivative $\frac{\mathrm{d}^2 u_{\text{pred}}}{\mathrm{d}(x_f^i)^2}$ can be determined using the chain rule equation (2).

The DE in this problem does not contain the terms $u(x)$ in it. Therefore, the Dirichlet boundary conditions cannot be applied using floating boundary handling (refer appendix B.5). Hence, the loss function is formulated as a multi-objective loss function appendix A. This loss function requires balancing the loss terms α_f, α_b. Therefore, two cases are compared: in the first case, we show the effect of unbalanced loss terms, where the weights of the total loss function are given with $\alpha_f = \alpha_b = 1$ (Case 1), and in the second case, the weights are constant factors and they are heuristically chosen as $\alpha_f = 10^{-1}$ and $\alpha_b = 10^3$ (Case 2). For these cases, a circuit with three qubits and a depth of seven blocks in the variational circuit appendix B.2 has been considered where the observables are chosen with the standard $\hat{\mathcal{C}} = \sum_j Z_j$.

The obtained solution for cases 1 and 2 together with the first and second derivative of the solution is visualized in figures 5(c), (e) and (f). The accuracy of the solutions is compared through the R^2 score using the term $1 - R^2$ (see figure 5(d)). The results comparison between cases 1 and 2 suggest that balancing weights are important while handling multi-objective loss function. However, the accuracy of the quantum ML circuit has trouble in stabilizing below 1%.

To further improve the accuracy of the given problem, we study the effect of more entangling layers in the variational circuit. Consider the variational circuit represented in figure 5(a) as the reference circuit (RC) corresponding to variational part of figure 1. A modified variational circuit with more entangling layers is represented in figure 5(b), which we refer to as an added entanglement circuit (AEC). With this modified circuit, we carry out another simulation use case (Case 3) with balanced weights of $\alpha_f = 10^{-1}$ and $\alpha_b = 10^3$.

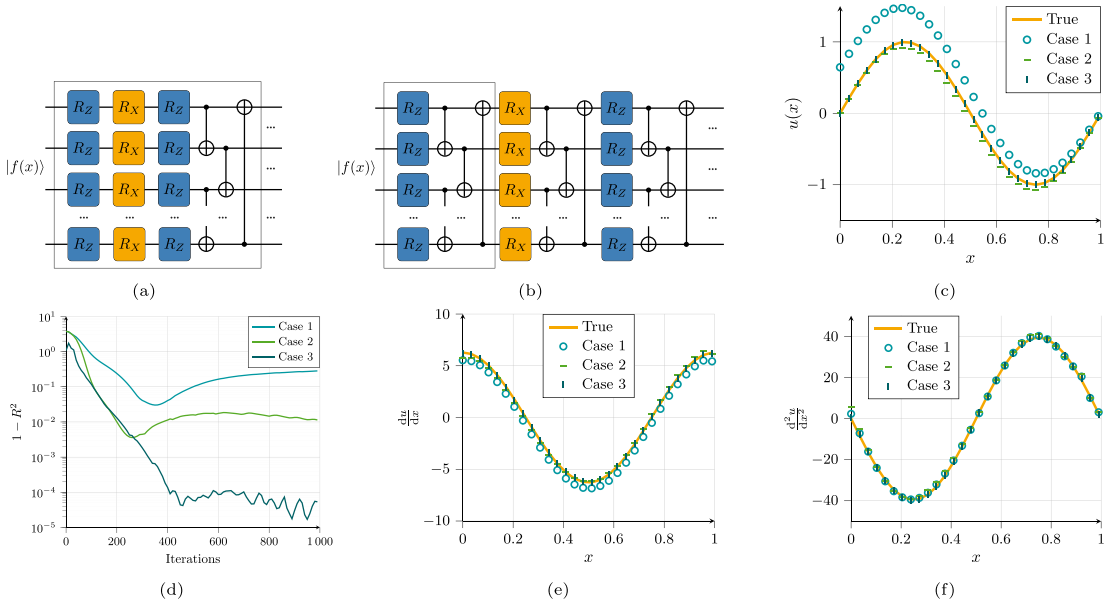

Figure 5. Solution of second-order DE (refer equation (22)) using constant weights for balancing multi-objective loss function. Visualization of (a) reference variational quantum circuit (refer figure 1) (referred to as RC). It is arranged as RZ, RX, RZ rotational layers followed by CNOT gates in cyclical order (b) modified variational circuit with added entangling layers in between rotational layers (referred to as AEC) (c). Solutions $u(x)$ obtained from three cases and their comparison with the true solution. Case 1 corresponds to employing of RC variational circuit, and balancing weights for multi-objective loss function $\alpha_f = \alpha_b = 1$. Case 2 corresponds to the RC variational circuit, and weights $\alpha_f = 10^{-1}$ and $\alpha_b = 10^3$. Case 3 corresponds to the AEC variational circuit, and weights $\alpha_f = 10^{-1}$ and $\alpha_b = 10^3$ (d). Comparison of accuracy in terms of $1 - R^2$ measurement over iterations among three cases, and checking of (e) first derivative $\frac{du}{dx}$ and (f) second derivative $\frac{d^2u}{dx^2}$ of solution after training.

For all cases and as in the previous DEs, an Adam optimizer was employed, where now the learning rate was specified at 0.01 for a total of 1000 iterations.

In the comparison between cases 2 and 3, the results suggest that adding entangling layers has enhanced the reachability of the circuit. It can improve the convergence and accuracy with the same number of variational parameters and their initial values. Therefore, entangling layers have played a key role in improving the accuracy while solving the second-order DE. While this worked for this example, [29] indicates that added entanglement hinders the optimization leading to a barren plateau. Hence, care must be taken while using larger circuits and emerging methods must be further explored to mitigate this issue [30–32].

To further explore the generalized case of variable balancing weights of the multi-objective loss function, an alternative approach based on SAPINNs [23] is incorporated (refer section 2.2). Based on the specifications provided in section 2.3, we define the mask functions for this problem as shown in figures 6(a) and (b).

We consider the modified variational circuit with added entangling layers (see figure 5(b)) and the same initial variational parameters from cases 1, 2, and 3 for a fair comparison. From the displacement, first, and second derivative graphs, both the polynomial and logistic masks can approximate the solution to a satisfactory level (see figures 6(c), (e) and (f)). The accuracy comparison in figure 6(d) shows that the simple constant weights (Case 3) were able to perform better. However, the results prove that the approach of self-adaptive weights is suitable for variational quantum circuits and can be used for various problems. Furthermore, as the complexity of the problem increases with a complex optimization landscape and more loss terms in the multi-objective loss function, the approach of self-adaptive weights will provide us with better hyperparameters to tweak as compared to repetitive trials with non-adaptive constant weights. Therefore, this approach might have more potential to save time in exploring hyperparameters and have a better convergence as compared to non-adaptive constant weights.

3.4. Second-order nonlinear DE

For the next problem, we consider a Duffing equation which is a nonlinear second-order DE given by,

$$\frac{d^2u}{dt^2} + \delta\frac{du}{dt} + \alpha u + \beta u^3 = \gamma\cos(\omega t). \tag{25}$$

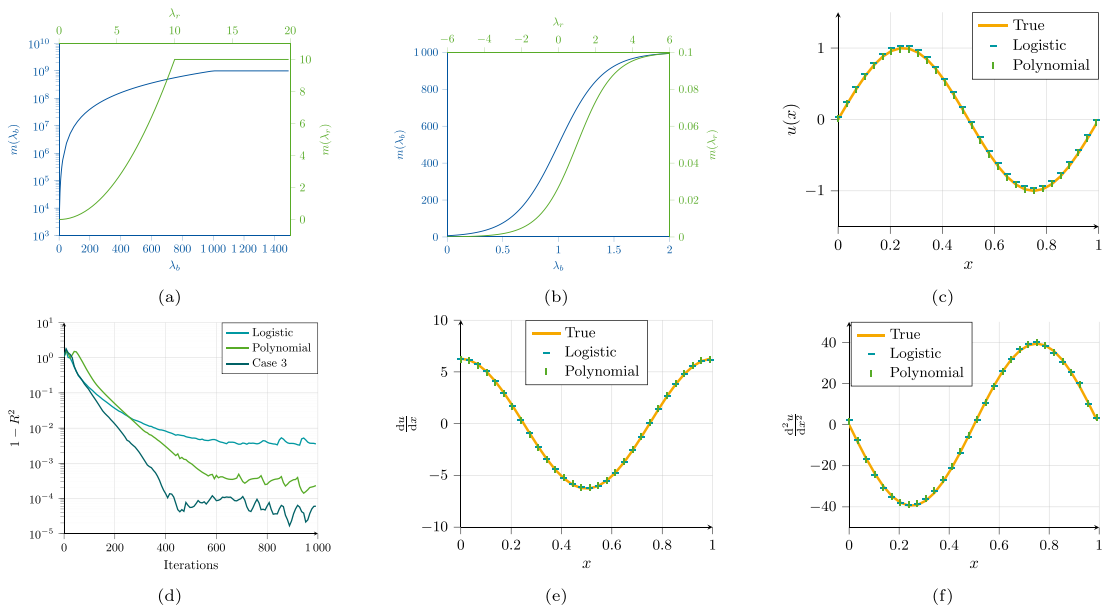

Figure 6. Solution of second-order DE (refer equation (22)) using self-adaptive weights approach. Visualization of (a) polynomial mask $m(\lambda)$ for residual (r) and for boundary (b), (b) logistic mask $m(\lambda)$ for residual (r) and for boundary (b), (c) solution $u(x)$ comparison between logistic and polynomial cases, (d) accuracy comparison in terms of $1-R^2$ measurement over iterations for logistic and polynomial cases. In addition, the accuracy curve from case 3 is also added here for reference. Testing of (e) first derivative $\frac{du}{dx}$ and (f) second derivative $\frac{d^2u}{dx^2}$ of predicted solution $u(x)$.

In this work, we have considered an example of this problem from [33], where the parameters are given by, $\alpha = 1, \beta = 1, \delta = 0.1, \gamma = 0.1, \omega = 0.4$. The initial conditions are given by $u(0) = 0$ and $\frac{du(0)}{dt} = 0$. The time evolution to solve the problem is chosen from $t = 0$ to $t = 20$. As a reference solution (also referred to as true solution), we employed an explicit Runge–Kutta method of fifth order [34]. For the quantum solution, the dependent variable is suggested to be not close to 1.0 due to singularity (refer section 2.1). Therefore, the dependent variable in the Duffing equation is scaled and solved within the interval $t = [0, l]$, where we choose the limit, $l = 0.9$. Consider the total time of the evolution as $T = 20$ and the scaled time variable as $\tau = \frac{t}{T}l$. Then the modified DE is given by,

$$\left(\frac{l}{T}\right)^2 \frac{d^2u}{d\tau^2} + \frac{l}{T}\delta\frac{du}{d\tau} + \alpha u + \beta u^3 = \gamma \cos\left(\frac{\omega T \tau}{l}\right). \tag{26}$$

The solution grid is chosen to be the discretization of 50 equidistant time intervals. Similar to the previous problem, the cost function is formulated as $C = \alpha_f \mathrm{MSE}_f + \alpha_b \mathrm{MSE}_b$ (see appendix A), where the weights $\alpha_f = 1$ and $\alpha_b = 1$. In this case, there was no need for balancing these two terms while their magnitude was found to be of similar order. The hyperparameters chosen for this problem are five qubits and the depth of eight blocks in the variational part of the quantum circuit (refer to appendix B.2). The observables are chosen to be $\hat{\mathcal{C}} = \sum_j Z_j$. To test the applicability of the proposed added entangling layers, we have considered two cases similar to the previous problem. The first case is where we chose the RC figure 5(a) and the second one is the added entangling circuit figure 5(b). For both cases, the initial variational parameters are the same for fair comparison. An Adam optimizer with a learning rate of 0.01 is employed with 300 iterations. After solving the problem at coarse 50 discretized points, the trained circuit is tested to predict the solution at 500 equidistant time steps for visualization of a smooth solution. This is one of the advantages of the spectral methods over finite difference methods, namely of having smooth interpolation. The results presented in the figures 7(a) and (b) show that the displacement and velocity are well-matched with the Runge–Kutta solution. We have also depicted the phase space plot in figure 7(c), which is velocity vs displacement, to visualize the resulting Duffing oscillation. Furthermore, the accuracy plot using the measurement $1-R^2$ (cf figure 7(d)) clearly shows that the added entangling circuit has better performance as compared to the RC.

3.5. Second-order PDE

As a final use-case, in the domain of PDEs, we consider a two-dimensional Poisson's equation [24] given by

$$\Delta u(x,y) = g(x,y) \text{ with } x, y \in [0, 0.9] \times [0, 0.9]\,, \tag{27}$$

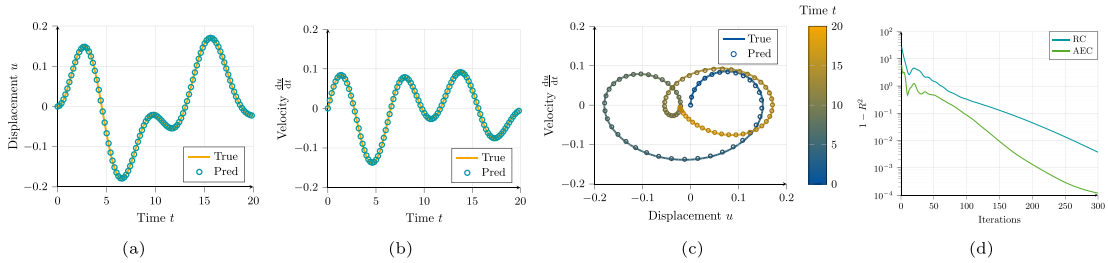

Figure 7. Solution of Duffing equation (refer equation (25)). Visualization of (a) displacement (u) vs time (t), (b) velocity ($\frac{du}{dt}$) vs time (t), (c) velocity vs displacement over the time evolution, also known as the phase-space plot of Duffing oscillation. The color bar on the right depicts the time (t). The accuracy comparison (d) in terms of $1 - R^2$ measurement over iterations for reference (RC) and added entanglement circuit (AEC).

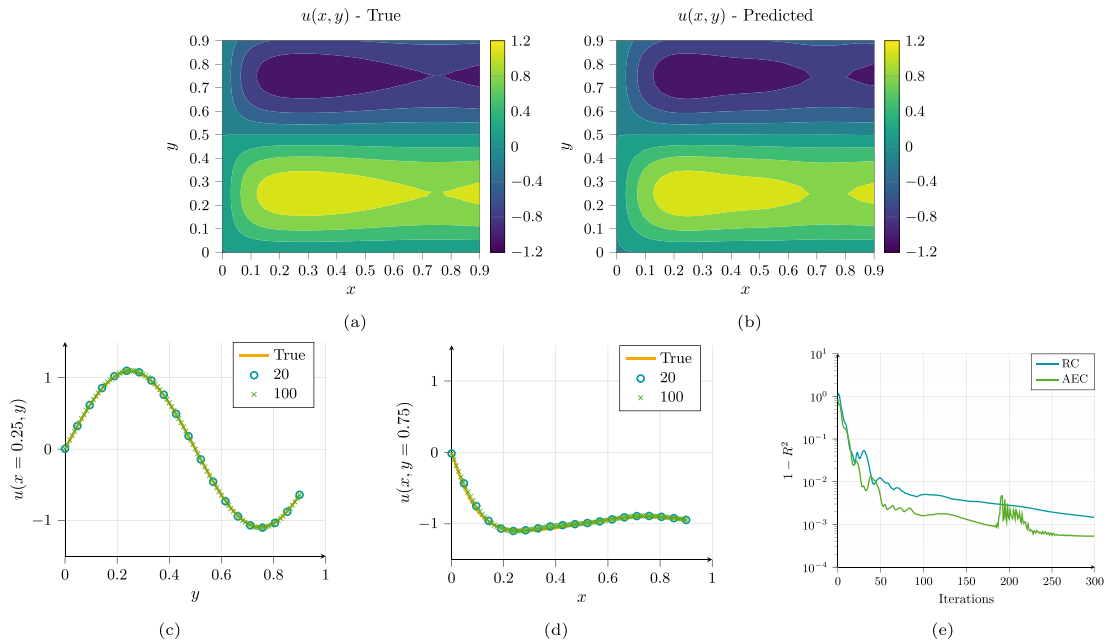

Figure 8. Solution of 2D Poisson's equation (refer equation (27)). Visualization of (a) true and (b) predicted solution $u(x, y)$ of Poisson's equation with color bar mapping the function values $u(x, y)$ to contour plot. Interpolation test for 20 and 100 equidistant points at (c) $x = 0.25$ and (d) $y = 0.75$, where the problem is originally solved at 30 equidistant points (e) the accuracy comparison in terms of $1 - R^2$ measurement over iterations for reference (RC) and added entanglement circuit (AEC).

where the homogenous Dirichlet boundary conditions and the right-hand side $g(x, y)$ are chosen in such a way that the exact solution is given by

$$u_{\text{true}}(x, y) = (0.1 \sin(2\pi x) + \tanh(10x)) \sin(2\pi y). \tag{28}$$

The quantum formulation of this problem is described in detail in section 2.4. The cost function for this problem consists of residual loss and boundary conditions. For calculating the residual loss, the domain for this problem is discretized into 30×30 equidistant points totaling 900 collocation points $N_f(x_f, y_f)$. The solution on the boundary is represented by N_b labeled data points $\{x_b^i, y_b^i, u_b^i\}$ and the Dirichlet boundary conditions are given by $u_b(x, 0), u_b(x, 0.9), u_b(0, y)$, and $u_b(0.9, y)$. Note that, we are examining a slightly altered problem domain than the usual boundary points $x = 1, y = 1$ due to computational costs and numerical instabilities in the differential operator leading to a modified boundary (see section 2.1). Finally, the cost function is formulated as $C = \alpha_f \text{MSE}_f + \alpha_b \text{MSE}_b$ (see appendix A), where the weights $\alpha_f = 10^{-1}$ and $\alpha_b = 10^3$. Adam optimizer with a learning rate of 0.1 is employed with 300 iterations.

The circuit architecture for this problem is chosen to be 3 qubits for variable x and 3 qubits for variable y with a depth of 7 blocks in the variational circuit (see appendix B.2). Two cases are considered for this problem for variational circuits, first RC and second AEC to study the applicability of added entangling layers. The predicted solution figure 8(b) from the AEC variational circuit is represented and compared with

the true solution figure 8(a). This approach has distinctly better convergence and accuracy when compared to the standard RC variational circuit, as illustrated in figure 8(e).

To further test the generalization performance of the trained circuit on the unseen data points, we have carried out an interpolation test within the solved domain. The domain is initially discretized to 30 equidistant points in between boundaries. Therefore, the interpolation test is conducted on a coarse grid (large steps) of 20 equidistant points and a fine grid (small steps) of 100 equidistant points. This test is performed on specific critical lines, such as $x = 0.25$ (see figure 8(c)) and $y = 0.75$ (see figure 8(d)). Based on these results, it can be observed that the trained circuit can interpolate and accurately predict the solution in unseen data points.

4. Discussion

Physics-informed quantum ML has an enormous potential for solving a diverse spectrum of problems related to DEs. A variety of approaches have been presented to solve DEs starting with the HHL algorithm, nonlinear variational solvers, quantum kernel methods, as well as using feature map encoding. In this work, the latter method is successfully combined with a self-adaptive variational quantum neural network to tackle challenging DEs. For initial value problems handled with floating boundary conditions, the proposed change in the measurement observables of the quantum circuit from the summation to the tensor product of Pauli-Z operators results in a remarkable accuracy improvement. The advantage of adding more entangling layers in the variational circuit is witnessed as an improvement in the accuracy while solving second-order DEs.

Initially, a highly nontrivial Riccati equation is solved, and a system of DEs is investigated comparing the results to existing solutions. In addition, a second-order linear DE is analyzed. The importance of balancing the multi-objective loss function is presented using constant balancing weights. To further explore the generalized case, an alternative approach based on self-adaptive weights is presented incorporating the optimization of balancing weights. Furthermore, a complex example of the nonlinear Duffing equation is analyzed. Lastly, a second-order 2D PDE is investigated. Therefore, a new quantum circuit is designed that can incorporate multiple independent variables.

Overall, the modified quantum circuits with added entangling layers lead to more accurate results in fewer epochs while solving second-order ordinary and PDEs. The enhancement in performance can be attributed to the fact that quantum circuits containing a larger number of entangling layers will result in a higher degree of expressibility. Consequently, the quantum states within the Hilbert space will be more thoroughly explored, leading to more precise outcomes. This work showcases the promise of quantum ML in addressing a variety of DEs, offering an innovative and efficient solution approach compared to classical methods. The adaptability of the proposed framework to different problem domains and its potential for further extensions, such as 3D PDEs and diverse applications e.g. in engineering or materials science, underscore the significance of advancing the field of quantum computing for solving real-world problems. Nevertheless, many questions remain regarding the capabilities of the presented work when applied to different problems such as nonlinear mechanics e.g. modeling finite-strain plasticity, solving the Navier–Stokes equation, or even coupled problems for fluid-structure interaction. A further question that still needs to be addressed is whether a potential quantum advantage can be achieved. Thus, the presented work is an initial starting point for further applications of QML presenting promising results to approach these problems.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgment

We would like to thank Mikhail Itskov and José Jesus for their helpful discussion and support. F M and A S received funding through the European Union's Horizon Programme (HORIZONCL4-2021-DIGITALEMERGING-02-10), Grant Agreement 101080085 (QCFD).

Author contribution

A S was responsible for conceptualization, methodology, implementation, original draft writing, and editing. R A and F M gave guidance and are responsible for editing and reviewing the draft.

Appendix A. Weighted multi-objective loss function

The DEs solved using variational quantum algorithms incorporating Chebyshev feature maps require minimizing the cost function which consists of the residual term for the DE itself and the boundary conditions.

Firstly, if the residual term for the DE is given by $f(x_f^i)$, the mean squared error MSE_f can be determined using N_f collocation points by

$$\mathrm{MSE}_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left(f\left(x_f^i\right)\right)^2. \tag{A.1}$$

Secondly, the boundary loss term MSE_b, which is only evaluated for all boundary points N_b is given by,

$$\mathrm{MSE}_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left(u_{\mathrm{pred}}\left(x_b^i\right) - u_b\left(x_b^i\right)\right)^2. \tag{A.2}$$

While solving the DE, it is required to minimize the total loss function by combining both residual and boundary loss terms. However, due to differences in the order of magnitude of these terms, it is beneficial to balance the loss terms in magnitude while optimizing. Therefore, a weighted approach for the total loss C can be formulated using

$$C = \alpha_f \mathrm{MSE}_f + \alpha_b \mathrm{MSE}_b, \tag{A.3}$$

where α_f and α_b are weights counteracting the imbalance of order between the residual and boundary loss terms.

Appendix B. Solving DEs using quantum chebyshev feature maps

For the quantum circuit shown in the figure 1, this section details the methodology employed in this work. As Kyriienko *et al* [21] stated, the objective is to approximate the solutions of the DEs as quantum circuits parameterized by a variable $x \in \mathbb{R}$ (or a collection of ν variables, $\mathbf{x} \in \mathbb{R}^\nu$). For brevity, a simplified single variable notation x is used for the generalized case of ν variables. The data is encoded into the quantum circuit using a *quantum feature map* $\hat{\mathcal{U}}_\phi(x)$, where *nonlinear* function of variables $\phi(x)$ are predefined. These quantum feature maps encode the data as amplitudes of the quantum state $\hat{\mathcal{U}}_\phi(x)|0\rangle$ from some initial state like $|0\rangle$.

Compared to other encoding techniques such as amplitude encoding which require access to the amplitudes, a quantum feature map encoding has several advantages. It can represent latent space encoding (multi-dimensional representation of compressed data) and it is controlled by gate parameters, mapping real parameter x to the corresponding variable value. Upon encoding, a vector of variational parameters θ are added by a variational quantum circuit $\hat{\mathcal{U}}_\theta$. These parameters are updated by a quantum–classical optimization loop, similar to classical ML.

The resulting state $|f_{\phi,\theta}(x)\rangle = \hat{\mathcal{U}}_\theta\hat{\mathcal{U}}_\phi(x)|0\rangle$ with optimized parameters contains the x-dependent amplitudes driven to represent the desired function. However, the resulting state is a quantum state and the desired function $f(x)$ is a classical function. Therefore, the real-valued function $f(x)$ can be brought out by calculating the expectation value of a predefined Hermitian cost operator $\hat{\mathcal{C}}$, such that

$$f(x) = \langle f_{\phi,\theta}(x)|\hat{\mathcal{C}}|f_{\phi,\theta}(x)\rangle. \tag{B.1}$$

Since solving DEs requires computing gradients $\frac{\mathrm{d}f(x)}{\mathrm{d}x}$, the quantum way of achieving this is by differentiating the quantum feature map circuit with

$$\frac{\mathrm{d}\hat{\mathcal{U}}_\phi(x)}{\mathrm{d}x} = \sum_j \hat{\mathcal{U}}_{\mathrm{d}\phi,j}(x), \tag{B.2}$$

where the index j runs through the individual quantum operations used in the feature map encoding. Here, derivatives are represented by the product derivative rule. When a quantum feature map is written by strings of Pauli matrices or any involutory matrix, the *parameter shift rule* can be used. Then the function derivatives are written as a sum of expectations with

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \frac{1}{2}\sum_j\left(\langle f^+_{\mathrm{d}\phi,j,\theta}(x)|\hat{\mathcal{C}}|f^+_{\mathrm{d}\phi,j,\theta}(x)\rangle - \langle f^-_{\mathrm{d}\phi,j,\theta}(x)|\hat{\mathcal{C}}|f^-_{\mathrm{d}\phi,j,\theta}(x)\rangle\right), \tag{B.3}$$

where $|f^{\pm}_{\phi,j,\theta}(x)\rangle$ is defined through the parameter shifting. The second-order derivative $\frac{d^2 f(x)}{dx^2}$ can be determined using four shifted terms for each generator. For simulators, automatic differentiation (AD) comes as handy in this situation. AD represents the function derivatives by an exact analytical formula using a set of simple computational rules, as opposed to numerical differentiation.

The solution of DEs is represented by constructing and defining the conditions for the quantum circuit, such that

$$F\left[\{d^m f_n/dx^m\}_{m,n}, \{f_n(x)\}_n, x\right] = 0, \tag{B.4}$$

where the functional F is provided by the problem for function derivatives of different order m and function/variable polynomials of varying degree n. For brevity, consider f as both a function and a vector of functions. Then this functional can be given as a task for optimization problem with loss function $\mathcal{L}_\theta[d_x f, f, x]$. In other words, this is a minimization problem $F[x]|_{x\to x_i}$ at points in the set $\mathbb{X} = \{x_i\}_{i=1}^M$, as well as taking boundary conditions into account. After the optimization, the updated angles

$$\theta_{\text{opt}} = \underset{\theta}{\text{argmin}}\left(\mathcal{L}_\theta\left[d_x f, f, x\right]\right) \tag{B.5}$$

are used to represent the solution

$$f(x)\,|_{\theta\to\theta_{\text{opt}}} \approx f(x)_{\text{desired}}\,. \tag{B.6}$$

In the following subsections, we discuss in detail the Chebyshev feature maps, variational quantum circuits, and so on.

B.1. Chebyshev feature maps

A nonlinear quantum feature map, namely, *Chebyshev feature map* changes the basis set of function representation by leveraging Chebyshev polynomials. They are defined as a single qubit rotation $R_{Y,j}(\phi[x])$, but with nonlinear function $\phi(x) = 2n\arccos x, n = 0, 1, 2, \cdots$, such that the encoding circuit is written as,

$$\hat{\mathcal{U}}_\phi(x) = \bigotimes_{j=1}^{N} R_{Y,j}(2n[j]\arccos x)\,. \tag{B.7}$$

Here, the coefficient $n[j]$ depends on the qubit position j. Using Euler's formula, the equation (B.7) can be rewritten as,

$$\begin{aligned} R_{Y,j}(\phi[x]) &= \exp\left(-\mathrm{i}\frac{2n[j]\arccos(x)}{2}Y_j\right), \\ &= \cos(n[j]\arccos(x))I_j - i\sin(n[j]\arccos(x))Y_j. \end{aligned} \tag{B.8}$$

To rewrite the above equation (B.8), consider the definition of Chebyshev polynomials [35]. They are determined by two sequences relating to cosine and sine functions, noted as $T_n(x)$ and $U_n(x)$.

Chebyshev polynomials of the first kind T_n are defined by,

$$T_n(\cos\theta) = \cos(n\theta). \tag{B.9}$$

Similarly, Chebyshev polynomials of the second kind U_n are defined by,

$$U_n(\cos\theta)\sin\theta = \sin((n+1)\theta). \tag{B.10}$$

Using the above relations, the equation (B.8) is further decomposed into unitary operation with matrix elements defined by degree-n Chebyshev polynomials of the first and second kind, such that,

$$R_{Y,j}(\phi[x]) = T_n(x)I_j + \sqrt{1-x^2}U_{n-1}(x)X_jZ_j. \tag{B.11}$$

In this work, a specific function $n[j] = j$ is chosen as it will expand the basis functions to the order of nth degree polynomial [21]. Thus, the updated feature map is given by

$$\hat{\mathcal{U}}_\phi(x) = \bigotimes_{j=1}^{N} R_{Y,j}(2j\arccos x)\,, \tag{B.12}$$

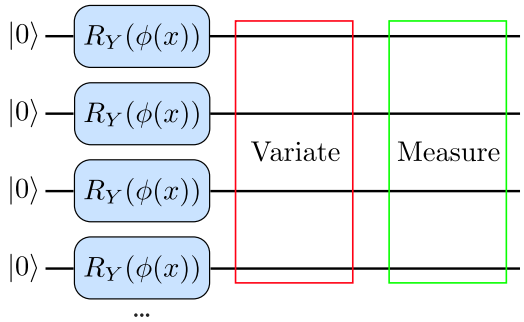

Figure B.9. Quantum Chebyshev feature map with single qubit rotations act at each qubit individually and are parameterized by a nonlinear function $\phi(x)$ of variable x. The variational circuit and measurement of the circuit are given by red and green blocks, respectively.

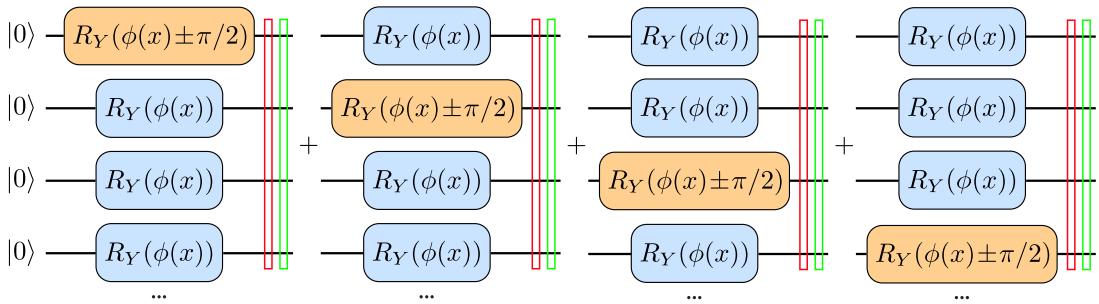

Figure B.10. Derivative of the quantum feature map. Differentiation with respect to x is followed by the chain rule, with the expectation value of the derivative written as a sum of separate expectations with shifted phases, repeated for each x-dependent rotation.

where the encoded variable expands with the number of qubits, making a tower-like structure of polynomials with increasing $n = j$. When the polynomials multiply with each other and morph between two kinds and their degrees, the basis set will become massive. This solves the problem of large expressibility without increasing the system size and number of rotations.

Consider the feature map circuit as in figure B.9. The derivative of the quantum circuit is calculated using the parameter shift rule (refer equations (B.2) and (B.3)) as shown in figure B.10.

B.2. Variational quantum circuit

Solving DEs as a quantum circuit means converging both derivatives and functions to the desired form. This requires manipulating the latent space basis function through the variational circuit $\hat{\mathcal{U}}_\theta$, which is referred to as a *variational quantum ansatz*. In this work, a well-known architecture, namely, **hardware efficient ansatz** is implemented.

Hardware efficient ansatz consists of layers of parameterized rotations, followed by layers of CNOT operations shown in figure B.11. A sequence of three layers of rotations $R_Z - R_X - R_Z$ are concatenated and parameterized by independent angles θ such that arbitrary single-qubit operations can be reproduced. CNOT gates act as an entangling layer. The rotations and CNOT gates together are referred to as blocks and this block is repeated for a depth of d times. The number of layers d is directly proportional to the circuit's expressive power (ability to represent arbitrary N-qubit unitary gates). However, as the parameters increase, the optimization will suffer with a problem known as **barren plateau**. The choice of ansatz is still an open question for improving trainability and requires further studies in this direction.

B.3. Observables

To get the output from quantum circuits, a Hermitian observable $\hat{\mathcal{C}}$ is used to measure the expectation. The scalar function $f(x)$ is then computed by expectation of $\hat{\mathcal{C}}$, written as, $\langle f_{\phi,\theta}(x)|\hat{\mathcal{C}}|f_{\phi,\theta}(x)\rangle$. Out of many available observables, the simplest is the magnetization of a single qubit j, $\langle Z_j\rangle$, which represents the function in range $[-1, 1]$. This choice requires rescaling for other intervals. Other options are total magnetization in the system $\hat{\mathcal{C}} = \sum_j Z_j$.

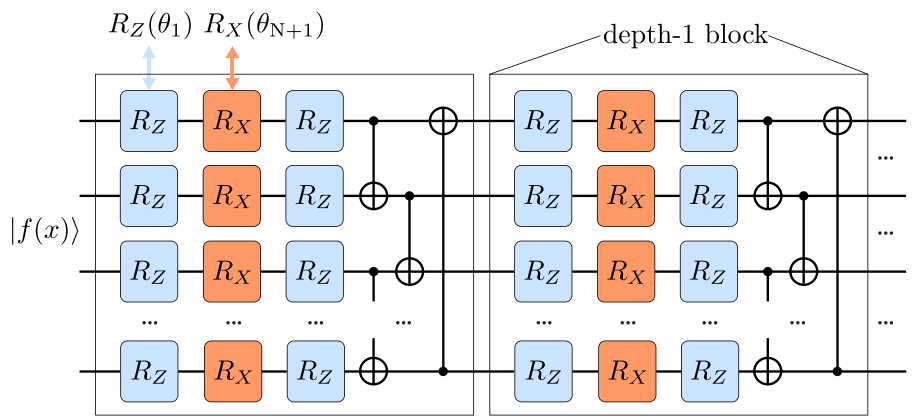

Figure B.11. Hardware efficient ansatz as a variational quantum circuit along with the representation of a single block/layer.

In addition to computing expectations, a weighted sum of observables is also possible, such as,

$$\hat{\mathcal{C}} = \sum_{\ell} \alpha_\ell \hat{\mathcal{C}}_\ell, \tag{B.13}$$

where $\alpha_\ell \in \mathbb{R}$ are trainable coefficients, and $\hat{\mathcal{C}}_\ell$ are usual observables as described above. These tunable coefficients α_ℓ are adjusted by optimizers such as gradient descent, Adam, etc. This is the beauty of hybrid quantum–classical algorithms.

B.4. Loss function handling

The loss function to solve DEs using quantum circuits is similar to classical PINNs. The classical optimizer updates the variational parameters such that the predicted function matches the desired function. This is achieved by reducing the distance between differential expression and zero evaluated at a set of collocation points. It is also required to satisfy the initial and boundary conditions. In other words, this resembles an optimization problem for a loss function of derivatives and functions evaluated at collocation points.

A generalized loss function parameterized by variational parameters θ is defined as,

$$\mathcal{L}_\theta\left[d_x f, f, x\right] = \mathcal{L}_\theta^{(\text{diff})}\left[d_x f, f, x\right] + \mathcal{L}_\theta^{(\text{boundary})}\left[f, x\right], \tag{B.14}$$

where $\mathcal{L}_\theta^{(\text{diff})}$ corresponds to differentials, and $\mathcal{L}_\theta^{(\text{boundary})}$ corresponds to boundary conditions. The differential loss is defined as,

$$\mathcal{L}_\theta^{(\text{diff})}\left[d_x f, f, x\right] = \frac{1}{M}\sum_{i=1}^{M} L\left(F\left[d_x f(x_i), f(x_i), x_i\right], 0\right), \tag{B.15}$$

where $L(a,b)$ is a function to describe the distance between the two arguments a and b. The loss is evaluated at M points and normalized by the size M. Functional F determines the DE described in the form $F[d_x f, f, x] = 0$. Here, the functional means it includes all DEs when the problem is a system, such that all equations are taken into account. For example, a Neumann boundary condition can also be included in the system. The boundary conditions especially, Dirichlet boundary conditions loss are defined by,

$$\mathcal{L}_\theta^{(\text{boundary})}\left[f, x\right] = \eta L\left(f(x_0), u_0\right), \tag{B.16}$$

which calculates the distance between the function value at the boundary x_0 and the given boundary value u_0. Similarly, the point x_0 here also can be the initial point instead of the boundary value. The factor η is referred to as the boundary pinning coefficient which regulates the weight of the boundary term in the optimization procedure. To emphasize the boundary precision, larger $\eta > 1$ can be used to prioritize.

The function L is for distance definition and can be defined in many ways. Popular one is a *mean squared error* (MSE) given as,

$$L(a,b) = (a-b)^2. \tag{B.17}$$

Additionally, the other distance choice is *mean absolute error* (MAE), defined as, $L(a,b) = |a-b|$. The choice of loss function affects the optimizer's performance on convergence. MSE works better by punishing the loss function harder with larger distances and lighter with smaller distances. MAE converges slowly, but once the training is close to the optimal solution, it can achieve higher accuracy than MSE.

B.5. Floating boundary handling

A simple way of handling Dirichlet boundary or initial conditions would be adding the boundary loss terms (equation (B.16)) into the total loss function (equation (B.14)) and trying to optimize both functions simultaneously. In multiple loss functions, trying to optimize one loss function will affect the performance of the other. However, if the DE has the term $f(x)$ in it, there is a possibility to solve the problem in a more efficient way known as **floating boundary handling**.

Floating boundary handling corresponds to recursively shifting the predicted solution based on the boundary or initial point. With this method, the boundary loss term can be removed from the total loss, rather it is encoded in the expectation of observable. As the function is parameterized to match a specific boundary, boundary information is contained within the function and its derivatives. In this way, the need for separate boundary loss terms can be eradicated. The updated function is described as,

$$f(x) = f_b + \langle f_{\phi,\theta}(x) | \hat{\mathcal{C}} | f_{\phi,\theta}(x) \rangle, \tag{B.18}$$

where $f_b \in \mathbb{R}$ is the parameter that is adjusted after each iteration step as,

$$f_b = u_0 - \langle f_{\phi,\theta}(x_0) | \hat{\mathcal{C}} | f_{\phi,\theta}(x_0) \rangle. \tag{B.19}$$

The solver will effectively find a function $\langle f_{\phi,\theta}(x) | \hat{\mathcal{C}} | f_{\phi,\theta}(x) \rangle$ that solves the DE shifted to any position, then being shifted to the desired initial condition as shown in equation (B.19). The advantage of this method is that the derivative loss term does not need to compete with the boundary loss. This also allows the optimizer to find the optimal angles and reduces the dependence on initial θ_{init}. This method can be generalized to multivariable problems which have an initial condition as a function of a subset of the independent variables.

ORCID iDs

Abhishek Setty https://orcid.org/0009-0001-4858-7985
Rasul Abdusalamov https://orcid.org/0000-0003-4988-4794
Felix Motzoi https://orcid.org/0000-0003-4756-5976

References

[1] Newton I 1774 Methodus fluxionum et serierum infinitarum, Opuscula mathematica, philosophica et philologica *History Sci.* **1** 1774
[2] Leibniz G 1858 Historia et origo calculi differentialis (1714) *Leibniz GW Math. Schriften* **7** 392–410
[3] Courant R, Friedrichs K and Lewy H 1928 Über die partiellen differenzengleichungen der mathematischen physik *Math. Ann.* **100** 32–74
[4] Galerkin B G 1915 Rods and plates series occurring in various questions concerning the elastic equilibrium of rods and plates *Eng. Bull. Vestnik Inzhenerov* **19** 897–908
[5] Ritz W 1909 Über eine neue methode zur lösung gewisser variationsprobleme der mathematischen physik
[6] Demirdzic I, Martinovic P and Ivankovic A 1988 Numerical simulation of thermal deformation in welded workpiece *Zavarivanje* **31** 209–19
[7] Raissi M, Perdikaris P, Karniadakis G E 2017 Physics informed deep learning (part i): data-driven solutions of nonlinear partial differential equations (arXiv:1711.10561)
[8] Raissi M, Perdikaris P and Karniadakis G 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686–707
[9] Cai S, Mao Z, Wang Z, Yin M and Karniadakis G E 2021 Physics-informed neural networks (pinns) for fluid mechanics: a review *Acta Mech. Sin.* **37** 1727–38
[10] Zhang E, Dao M, Karniadakis G E and Suresh S 2022 Analyses of internal structures and defects in materials using physics-informed neural networks *Sci. Adv.* **8** eabk0644
[11] Lloyd S, Mohseni M and Rebentrost P 2014 Quantum principal component analysis *Nat. Phys.* **10** 631–3
[12] Lloyd S 2010 Quantum algorithm for solving linear systems of equations *APS March Meeting Abstracts* vol 2010 pp D4–002
[13] Lapworth L 2022 A hybrid quantum-classical CFD methodology with benchmark hhl solutions (arXiv:2206.00419)
[14] Tosti Balducci G, Chen B, Möller M, Gerritsma M and De Breuker R 2022 Review and perspectives in quantum computing for partial differential equations in structural mechanics *Front. Mech. Eng.* 75 914241
[15] Lubasch M, Joo J, Moinier P, Kiffner M and Jaksch D 2020 Variational quantum algorithms for nonlinear problems *Phys. Rev.* A **101** 010301
[16] Umer M, Mastorakis E, Evangelou S, Angelakis D G 2024 Nonlinear quantum dynamics in superconducting nisq processors (arXiv:2403.16426)

[17] Paine A E, Elfving V E and Kyriienko O 2023 Quantum kernel methods for solving regression problems and differential equations *Phys. Rev.* A **107** 032428
[18] Oz F, San O and Kara K 2023 An efficient quantum partial differential equation solver with chebyshev points *Sci. Rep.* **13** 7767
[19] Demirdjian R, Gunlycke D, Reynolds C A, Doyle J D and Tafur S 2022 Variational quantum solutions to the advection–diffusion equation for applications in fluid dynamics *Quantum Inf. Process.* **21** 322
[20] Mitarai K, Negoro M, Kitagawa M and Fujii K 2018 Quantum circuit learning *Phys. Rev.* A **98** 032309
[21] Kyriienko O, Paine A E and Elfving V E 2021 Solving nonlinear differential equations with differentiable quantum circuits *Phys. Rev.* A **103** 052416
[22] Williams C A, Paine A E, Wu H-Y, Elfving V E, Kyriienko O 2023 Quantum chebyshev transform: mapping, embedding, learning and sampling distributions (arXiv:2306.17026)
[23] McClenny L D and Braga-Neto U M 2023 Self-adaptive physics-informed neural networks *J. Comput. Phys.* **474** 111722
[24] Kharazmi E, Zhang Z and Karniadakis G E 2021 Hp-vpinns: Variational physics-informed neural networks with domain decomposition *Comput. Methods Appl. Mech. Eng.* **374** 113547
[25] Bergholm V *et al* 2018 Pennylane: automatic differentiation of hybrid quantum-classical computations (arXiv:1811.04968)
[26] Bogacki P and Shampine L F 1989 A3 (2) pair of runge-kutta formulas *Appl. Math. Lett.* **2** 321–5
[27] Hindmarsh A C 1983 Odepack, a systemized collection of ode solvers *Sci. Comput.* **1** 55–64
[28] Petzold L 1983 Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations *SIAM J. Sci. Stat. Comput.* **4** 136–48
[29] McClean J R, Boixo S, Smelyanskiy V N, Babbush R and Neven H 2018 Barren plateaus in quantum neural network training landscapes *Nat. Commun.* **9** 4812
[30] Kulshrestha A and Safro I 2022 Beinit: avoiding barren plateaus in variational quantum algorithms *2022 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* (IEEE) pp 197–203
[31] Larocca M, Czarnik P, Sharma K, Muraleedharan G, Coles P J and Cerezo M 2022 Diagnosing barren plateaus with tools from quantum optimal control *Quantum* **6** 824
[32] Mele A A, Mbeng G B, Santoro G E, Collura M and Torta P 2022 Avoiding barren plateaus via transferability of smooth solutions in a hamiltonian variational ansatz *Phys. Rev.* A **106** L060401
[33] Salas A H 2022 An elementary solution to a duffing equation *Sci. World J.* **1** 2357258
[34] Dormand J R and Prince P J 1980 A family of embedded runge-kutta formulae *J. Comput. Appl. Math.* **6** 19–26
[35] Mason J C and Handscomb D C 2002 *Chebyshev Polynomials* (CRC Press)