Array 18 (2023) 100287

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array

General implementation of quantum physics-informed neural networks

Shashank Reddy Vadyala [a,*], Sai Nethra Betgeri [a]

[a] Department of Computational Analysis and Modeling, Louisiana Tech University, Ruston, LA, United States

ARTICLE INFO

Keywords:
Physics
Machine learning
Quantum computing
Physics informed neural network

ABSTRACT

Recently, a novel type of Neural Network (NNs): the Physics-Informed Neural Networks (PINNs), was discovered to have many applications in computational physics. By integrating knowledge of physical laws and processes in Partial Differential Equations (PDEs), fast convergence and effective solutions are obtained. Since training modern Machine Learning (ML) systems is a computationally intensive endeavour, using Quantum Computing (QC) in the ML pipeline attracts increasing interest. Indeed, since several Quantum Machine Learning (QML) algorithms have already been implemented on present-day noisy intermediate-scale quantum devices, experts expect that ML on reliable, large-scale quantum computers will soon become a reality. However, after potential benefits from quantum speedup, QML may also entail reliability, trustworthiness, safety, and security risks. To solve the challenges of QML, we combine classical information processing, quantum manipulation, and processing with PINNs to accomplish a hybrid QML model named quantum based PINNs.

1. Introduction

The first ideas for quantum information processing were published in the 1970s [1]. The theoretical foundations of QC date back to the 1980s when Russian and American physicists first conceptualized the quantum computer field [2]. The appeal of these early contributions was that they showed that quantum bits could carry more information than classical bits so that QC could be expected to solve specific complex problems much faster than classically possible. It took about another decade until the first quantum algorithms were conceived to exhibit these hypothesized advantages when running on a quantum computer [3,4]. However, when these algorithms were presented, they were mere theoretical exercises in "pen and paper programming" since quantum computers on which they could have been implemented did not yet exist. Nevertheless, Grover's amplitude amplification algorithm convincingly demonstrated that quantum computers could exhaustively search unstructured lists quadratically faster than digital computers [5]. Shor's celebrated quantum algorithm for large integer factorization revealed exponential speedup over classical approaches [6]. The latter sparked broader attention to the idea of QC because Shor's discovery implied that typical data encryption schemes would no longer be secure if working quantum computers were to become a technical reality.

The advance of quantum information technologies introduces various uses that might change the future of science and technology. Before the Schrödinger equation was first published [7], it was unclear and specific how quantum physics could affect information technology. However, in the last few decades, rapid quantum development started to take a swing [8,9]. In the meantime, ML has become an important and modern tool for advanced data usage [10–14]. NNs were introduced in modern information technologies as a vital tool for ML. In and of itself, ML is not a new idea; it was initially conceptualized by Turing and had its first boom period in the 1980s after algorithms for training Artificial Neural Networks (ANNs) became available. Till today, there have been advanced ways of using NNs to help and solving problems, such as in quantum chemistry [15–17], pattern recognition [18,19], sequence recognition [20,21], data mining [22,23], finance [24–26], medical diagnosis [27–31], visualization [32,33], solving PDEs [34,35], ocean modeling and coastal engineering [36], geomorphology [37], simulating properties of many-body open quantum systems [38], etc. Although their ML methods exist for very capable learning systems of comparatively moderate sizes [39–41], the capability of a learning system to exhibit human-like performance seems to be predicated on the fact that its size far exceeds traditional system sizes. In other words, the underlying mathematical models require more flexibility and, thus, more adjustable parameters than have been considered historically. For instance, the adjustable parameters of an artificial neural network are its synaptic weights, and modern deep networks often come with hundreds of billions of such weights (see, e.g. Ref. [42]). Given numbers like these, training modern learning systems is a resource-intensive endeavor that easily translates to several hundred GPU years of computation time [42].

* Corresponding author.
E-mail address: vadyala.shashankreddy@gmail.com (S.R. Vadyala).

https://doi.org/10.1016/j.array.2023.100287
Received 6 December 2022; Received in revised form 26 February 2023; Accepted 8 March 2023
Available online 16 March 2023

Current ML has reached a point where practical feasibility and success are conditioned on access to high-performance computing hardware. Against these circumstances, it is not surprising that ML researchers are beginning to look at another technology that has recently seen substantial progress, namely Quantum Information Processing (QIP) [43–45]. Indeed, since many of the fundamental problems in ML demand optimization or search problems of the kind QC can excel at, research on QML is noticeably intensifying. The corresponding literature has rapidly grown, and the first textbooks and tutorials have become available [44–46]. Moreover, more and more quantum coding challenges are being held that specifically focus on aspects of ML, and there are recognizable efforts by industrial players to get machine learners involved in QC research. Given these developments, experts predict that QML on reliable, large-scale quantum computers will soon facilitate demanding learning tasks and even put problems into reach which are intractable on digital computers. If these predictions materialize, QML technology will further accelerate AI progress, likely leading to novel marketable products and commercial solutions impacting economies and societies [47].

The most fundamental and important component of QML models is Parameterized Quantum Circuits (PQCs). These entail a series of unitary gates working on the quantum data states, some of which have free parameters θ and which will be educated to figure out the issue. PQCs and NNs are conceptually similar, and this comparison may be made precisely because PQCs can technically incorporate conventional NNs. Due to this, several PQCs types are now referred to as Quantum Neural Networks (QNNs) by researchers. In the following, we'll use the word "QNNs" because it's commonly used anytime a PQCs is utilized in a data science application. All three of the QML paradigms use QNNs. For instance, the QNN's objective in a supervised classification assignment is to map the states of the various classes to distinct areas of the Hilbert space. Additionally, a clustering job is translated into a MAXCUT issue in the unsupervised learning scenario and resolved by training a QNNs to maximize the distance between classes. Given its present state, a learning agent's optimum course of action may be determined using a QNNs as the Q-function approximator in the reinforced learning task. In the case of supervised learning, we are interested in generating precise predictions about (generalizing to) previously unobserved data in addition to learning from a training dataset. This minimizes training and prediction errors, often depending on the former. Let's now look at prediction error, also known as generalization error, which has recently been investigated for QML [48,49]. Formally stated, this error gauges how effectively a trained QML model works with unobserved data. The training error and complexity of the learned model both affect prediction error. If the training mistake is substantial, the prediction error is frequently considerable. The prediction error is probably still considering if the trained model's complexity is significant, but the training error is minor. Only when the training error is modest, and the trained model's complexity is moderate (i.e., sufficient to be less than the training data size) [14,35] is the prediction error small. Due to a lack of transparency and trust, QML may potentially have hazards in addition to possible advantages since there is a gap between the demands of users and the black-box nature of these models [50].

To solve the black-box character in QML models, we introduce an implementation of PINNs to QC machines. PINNs was introduced as NNs trained to complete supervised learning tasks while adhering to any given physical rules specified by generic nonlinear PDEs in a study produced by Raissi, Perdikaris, and Karniadakis [51] and paper meant a novel perspective in which NNs could benefit from knowing physical laws in the shape of PDEs. This synthesis of NNs and knowledge from physics opened a newly evolving area of research with implementations like fluid mechanics [52,53], cardiac activation mapping [54], PDEs [55], power systems [56], velocity inversion [57], myocardial perfusion MRI quantification [58], advection equation [59], etc. Utilizing high-performance frameworks, like TensorFlow [60] or PyTorch [61], created expressly for executing ML workloads effectively, is one of the

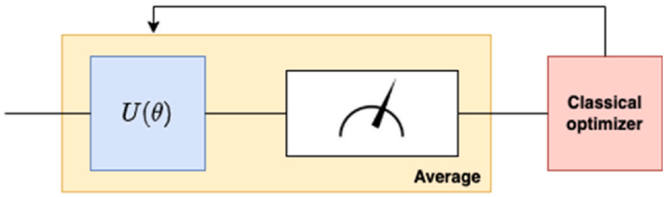


Fig. 1. An example of the QML design process: A classical optimizer uses a vector of parameters to optimize a parametrized θ quantum circuit with a predetermined design based on input data and output measurements. As we will see in this book, a unitary matrix $U(\theta)$, depending on the vector, determines how the parametrized quantum circuit operates. The block denoted by a gauge sign corresponds to a quantum circuit in classical information. Since the conversion is unpredictable, the measurement outputs are normally averaged before being sent to the traditional optimizer.

computational advantages of employing PINNs approaches for scientific computing from a strategic standpoint. However, because of the quantum mechanical properties of the machine, we shall consider some generalizations that produce, for example, using complex numbers and operator theory, as well as physical limitations when getting the results and making the interpretations. Note that the implementation will differ for some elements compared to that on a classical computer; not only physically but also the general form of information processing will change, as we will see in the following sections. The number of implementation cases of PINNs has risen almost continuously. We see the implementation of PINNs on quantum computers as a natural consequence that will help predict and solve complex data configurations and constellations [62]. scientists have been dealing with the Differentiable Quantum Circuits (DQCs) approach. Thus, this paper will consider the general cost function and Hamiltonian molding. The one-dimensional Schrödinger equation is used for quantum based PINNs evaluation.

2. Background

2.1. Quantum Machine Learning

Traditionally, QC algorithms have been developed manually, supposing the existence of fault-tolerant quantum processors that can consistently handle many qubits and quantum operations, also referred to as quantum gates. The fundamental building block of quantum information and computation, or qubit, is analogous to a bit in conventional computers. Quantum computers nowadays use a few tens of qubits with noise- and imperfection-prone quantum gates. A new alternative design paradigm called QML is being developed specifically for today's Noisy Intermediate-Scale Quantum (NISQ) computers. The strategy adheres to a two-step paradigm reminiscent of traditional ML. It involves fixing a priori, potentially generic, parametrized design for the quantum gates defining a quantum algorithm, followed by tuning the parameters of the gates using classical optimization. In QML, the quantum algorithm is described as a quantum circuit, designated as $U(\theta)$, whose constituent quantum gates carry out operations that rely on a vector θ of free parameters. This definition is illustrated in greater detail in Fig. 1. Data and measurements of the quantum state generated by the quantum circuit are supplied into a classical processor. The traditional optimizer modifies the vector θ to minimize some cost functions the designer provides.

The QML architecture, which assumes fault-tolerant QC, provides several potential advantages over the conventional method of developing quantum algorithms by hand.

- By incorporating the quantum computer into the optimization process, the classical optimizer may use measurements of the QC output to directly account for the shortcomings and constraints of quantum operations.

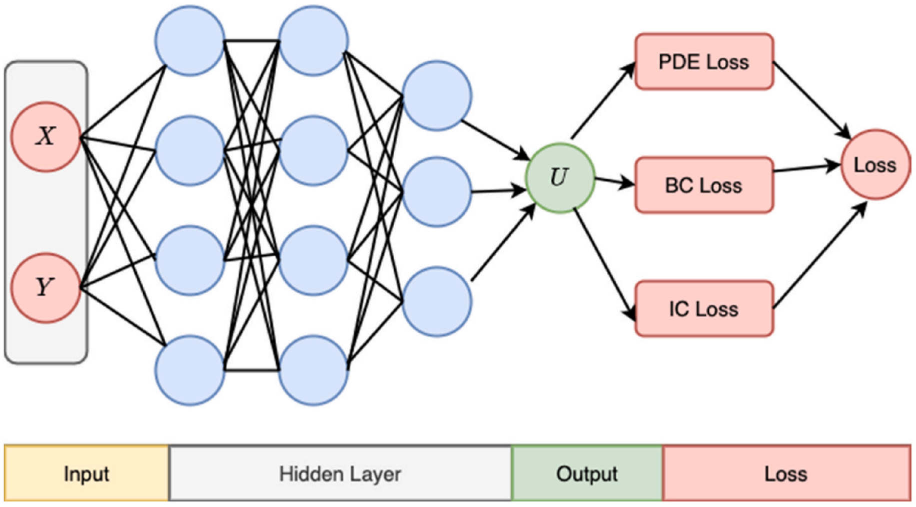


Fig. 2. A schematic of PINNs for solving general PDEs.

- Assume that the classical optimizer and parametrized quantum algorithm are both suitably powerful. In that situation, the methodology may automatically create effective quantum algorithms that would have been challenging to tune manually using conventional formal approaches.

2.2. *Physics informed neural network*

The paper by Raissi et al. [51] introduced a simple yet valuable way of incorporating knowledge of physical processes into computational learning. As data and computing resources have grown over the recent past, and the fields of ML and data science are evolving rapidly, it means reframing results for various spheres of science. Nonetheless, the cost of analyzing complex systems (in physics, biology, etc.) and data purchase is highly priced and forces us to make decisions and conclusions with incomplete information. The useful information used in the ML processes for complex systems is incorporated into laws of nature, often embodied in PDEs or some other experimentally approved results. Especially time-dependent dynamics that govern real processes have a primary place in modern ML. Including PDEs in the core of learning gives regularization and restriction to a range of justifiable solutions.

Moreover, integrating information constellations will result in maturing of algorithm information and reducing the time for data convergence toward a legitimate solution. There are examples of successful implementations of PINNs in mathematical physics [63,64]. In Ref. [51], the approach was taken by applying a deep neural network (DNNs) and leveraging the operational capacity as function approximators. The significant advantage comes from the surface as we realize that nonlinear problems can be attempted without any linearizations or prior assumptions. Automatic differentiation is exploited [65] to differentiate neural networks (with respect to input data) and model parameters to create PINNs. In that sense, PINNs respect, in general, symmetries, conservation laws, and invariances that emerge from laws of physics that dictate given data structures (as dictated with PDEs). This powerful but simple method gives us the potency to deal with many problems in computational physics and similar scientific fields. Let us consider parametrized and nonlinear PDEs of the general form. Where $x \in \Omega$ and $t = [0,T]$. Also, $u(t,x)$ represents the latent solution, $\mathcal{N}$ is a parametrized nonlinear operator (with λ), and Ω is a subset of $\mathbb{R}^D$. This mathematical setup encapsulates and covers a diverse sphere of problems in mathematical physics, like conservation laws, diffusion processes, kinetic equations, etc. The authors in Ref. [36] explain solving Burgers and Schrödinger equations. For continuous-time models, we define the function.

$$f = u_t + \mathcal{N}[u] \tag{2}$$

and approximate $u(t,x)$ by a DNNs. That premise, together with equation (2), gives a PINNs f. u and f Mutual parameters learned can be done by mean squared error loss minimization.

$$M = M_f + M_u \tag{3}$$

$$M_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left(t_f^i, x_f^i\right)\right|^2 \tag{4}$$

$$M_u = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|u\left(t_f^i, x_f^i\right) - u^i\right|^2 \tag{5}$$

Note that $\{t_f^i, t_f^i, u^i\}_{i=1}^{N_u}$ represents initial and boundary data for u and $\{t_j^i, t_j^i\}_{i=1}^{N_u}$ stipulate collocation data for. For discrete-time models, the RungeKutta method is often used [49], which is correlative to continuous models. In conclusion, classical numerical methods, like the RungeKutta time-stepping method, can exist side-by-side with DNNs and provide a helpful hunch in constructing structured predictive algorithms. A schematic of PINNs is shown in Fig. 2.

3. Method

3.1. *Overview of PINNs on quantum computers*

The theory of quantum information is based on the axioms of quantum mechanics and some additional rules that better describe the information processing processes. The primary description of the theory is done by introducing five famous theorems: the no-teleportation theorem, the no-cloning theorem, the no-deleting theorem, the no-broadcast theorem, and the no-hiding theorem. All these theorems limit us in the attempt to manipulate and use information encoded in quantum systems. However, by using some sophisticated algorithms with good laboratory equipment, one can get close to the goal of using quantum information as an advantageous way of processing information. Advancements in ML combined with the Universal Approximation Theorem [66] have made it possible to approximate functions using ANNs. This enabled the development of physics-informed ML [5] and novel techniques for solving differential equations representing the approximate solution using ANNs as the Universal Function Approximator (UFA). The optimization of PINNs is achieved by minimizing a loss function constructed by PDEs that encode the physical principles of a system. Specifically, the loss function may consist of an

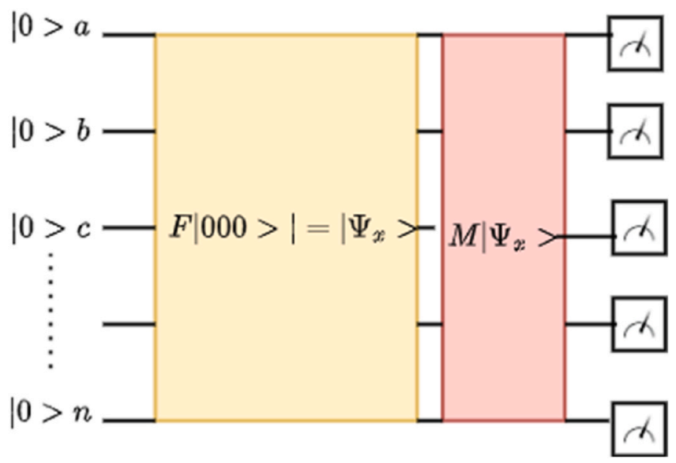


Fig. 3. Overview of QNNs architecture. The input is encoded into a high-dimensional quantum information system (qubits) on which a feature map F is applied. A variational model M then evolves the collective state. In the end, a measurement is performed, and the outcome is processed in the next cycle to extract the model's output.

equation-driven and a data-driven component [67]. The former depends only on the neural solutions and their derivatives with respect to the inputs; the derivatives are calculated using automatic differentiation (autographed). The latter can compare neural predictions to ground truth data or impose physical laws. Using PINNs to solve PDEs offers more flexibility (for example, incorporating data) and generality (for example, high-dimensional parameterized PDEs) compared to classical techniques. PINNs also led to the developing of quantum-ready algorithms to solve PDEs, employing QNNs, or 'Differentiable Quantum Circuit' (DQC) [62] as the UFA, offering an expanded latent space for representing expressive functions. For the sake of making a mathematical model for implementing PINNs on QC machines, we must introduce classically inspired quantum tools, which apply to quantum computers by analogy, as shown in Fig. 3. Note that not every element of the process of realization will be analog since quantum information is fundamentally different from classical information. Therefore, information processing will, in a lot of real-life cases, use the help of classical computers, in so-called hybrid models.

A quantum based PINNs expands the quantum neural network. An overview of the quantum PINNs process and resource use is shown in Fig. 4. We encode the collocation point as a real-valued displacement of the vacuum state, which is the lowest energy Gaussian state with no displacement or squeezing in phase space, as the initial step of the quantum-based PINNs. The simplicity of encoding the input (the collocation point) into the QNNs without the need for normalization is one of the key benefits of employing QNNs. The qumode flows into the quantum based PINNs after the first displacement encodes the collocation point.

3.2. Model of implementation

To successfully translate the PINNs structure and mechanism to a QC machine, the formalism described by Eq (1) must be mapped and reformed as a new formalism, but with consistency to the classical one. In doing so, let us look closely at Eq (1): the left-hand side has a term u_t That indicates the time-dependent dynamics of a system that is being observed. Indeed, many physical processes have the shape of PDEs; one of them is the time-dependent Schrödinger equation. We can derive this by implementing the evolution operator shown in Eq (5) on an arbitrary state $|\psi\rangle$, which results in Eq (6).

$$\widehat{U} = exp\left(-\frac{i\widehat{H}t}{h}\right) \tag{6}$$

$$\frac{\partial}{\partial t}\Psi + \frac{i}{h}H\Psi = 0 \tag{7}$$

By adjusting the notation and denoting $\frac{\partial}{\partial t}$ with $|\psi_t\rangle$

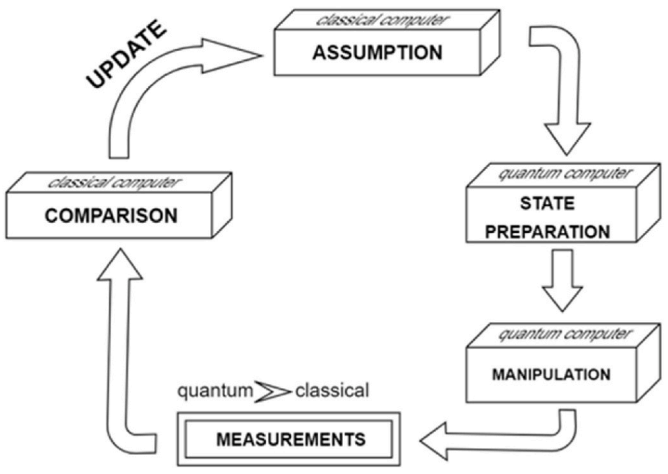


Fig. 5. The proposed QNNs realization scheme. First, the assumptions are prepared by a classical computer which assists in preparing quantum states on a quantum computer. The prepared quantum states are manipulated and measured in the selected basis set (quantum to classical information approximative conversion). Obtained results are compared, the cost function is minimized, and the parameters are updated for the next cycle.

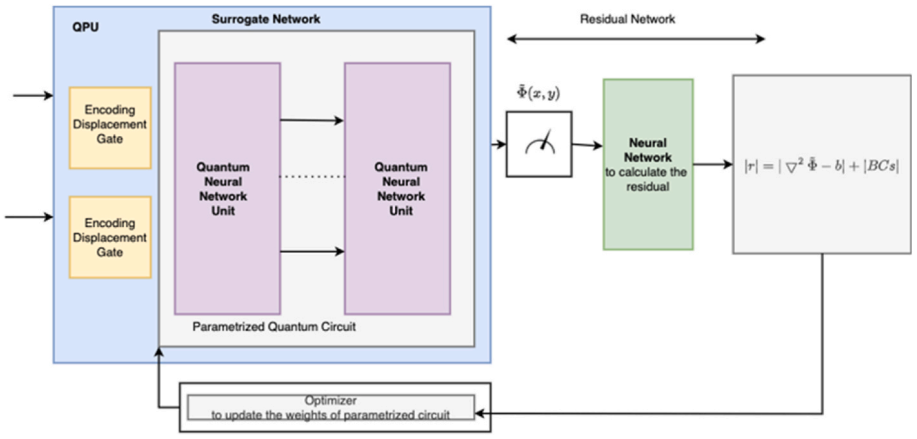
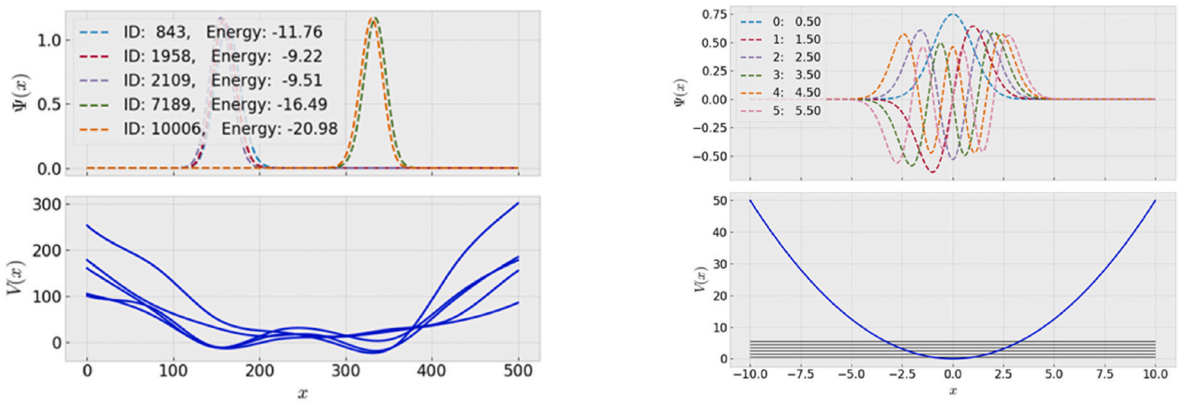

QPU
Surrogate Network
Encoding Displacement Gate
Encoding Displacement Gate
Quantum Neural Network Unit
Quantum Neural Network Unit
Parametrized Quantum Circuit
Residual Network
$\bar{\Phi}(x,y)$
Neural Network to calculate the residual
$|r| = |\nabla^2\bar{\Phi} - b| + |BCs|$
Optimizer to update the weights of parametrized circuit

Fig. 4. Shows a workflow model for quantum based PINNs. The input of the quantum-based PINNs is the values of the collocation points, which are represented as a displacement (only with the real part). The parametrized quantum circuit comprising several linked QNNs units is how the PINNs surrogate/approximator network is expressed. On the CPU, automated differentiation is used to calculate the PINNs residual function. Some optimizers might need a quantum computer to evaluate the cost function.



(a) Outlier data (b) Training dataset

Fig. 6. (a) Simulated data for varying $V(r)$ and $\psi(x)$ having noise. (b) Training data for quantum based PINNs.

$$|\psi_t\rangle \frac{i}{\hbar}\widehat{H}|\psi\rangle = |0\rangle \tag{8}$$

Eq (5). looks indeed like Eq (1), where the role of $\mathcal{N}$ is taken by the quantum system Hamiltonian $\widehat{H}$. Indeed, one can shape $\widehat{H}$ in many arbitrary ways if the arbitrary operator $\mathcal{N}_{quantum}$ obeys the fact $\widehat{H} = \widehat{H}_{\dagger}$. This paper proposes a classically inspired quantum algorithm that enables us to manipulate and process quantum information with tools for classical information processing. The reason for complete quantum algorithm avoidance is that information could be easily damaged or lost while storing and comparing due to quantum interference with the reservoir [68]. We will use classical tools for measuring quantum information in the assumption-making and comparison processes. However, the preparation of states and manipulation are tasks left to the quantum machine. A basic outline of the scheme is given in Fig. 5.

One must design a hamiltonian that fits a given problem to update parameters. In doing so, we must be careful about which quantum hardware platform to use. Superconducting qubits look most promising regarding pseudo-arbitrary Hamiltonian shaping [69]. Moreover, one needs to define appropriate cost functions for parameter optimization. To best define parameters, we must approximate the wave function ψ - or, in QC: the state vector $|\psi\rangle$. After enough shots of measurements, say n, a test function φ_n must be most like the given state ψ (and consequently ψ_t). Let us define the following to be an accessory function:

$$f_{\varphi,\psi} = \langle\varphi|\psi_t\rangle + \frac{i}{\hbar}\langle\varphi|\widehat{H}|\psi\rangle \tag{9}$$

The inner bracket products are also known as fidelities. One can see discussions on fidelity function assortment [70]. For Eq (6) to be a valid MSE-like function, we want it to be real:

$$F = ff^* = |\langle\varphi|\psi_t\rangle|^2 + \frac{1}{\hbar^2}|\langle\psi|\widehat{H}|\varphi\rangle|^2 + \frac{i}{\hbar}\left[\langle\psi_t|\varphi\rangle\langle\varphi|\widehat{H}|\psi\rangle - \langle\psi_t|\varphi\rangle^{\dagger}\langle\varphi|\widehat{H}|\psi\rangle^{\dagger}\right]$$

Eq (10)

Note that to compute Eq (7), one needs to derive four values, two of which are generally complex: $\langle\varphi|\widehat{H}|\psi\rangle$ and $\langle\psi_t|\varphi\rangle$.We then divide F into

$$F_R = |\langle\varphi|\psi_t\rangle|^2 + \frac{1}{\hbar^2}|\langle\psi|\widehat{H}|\varphi\rangle|^2 \tag{11}$$

$$F_I = \frac{i}{\hbar}\left[\langle\psi_t|\varphi\rangle\langle\varphi|\widehat{H}|\psi\rangle - \langle\psi_t|\varphi\rangle^{\dagger}\langle\varphi|\widehat{H}|\psi\rangle^{\dagger}\right] \tag{12}$$

Note that for real-state vectors, it automatically follows that $F_I = 0$, and we call it the phase cost function. Also, in Eq (11), the sign has changed, for the simple fact that when calculating $\mathrm{F_R}$ apart from F, the Schrödinger equation must be satisfied. F_R is set to be determined after all measurements occur, and the collected data goes through a comparison process on classical computers. However, Eq (12) cannot be easily approximated to classical information and requires further manipulation on a quantum computer to be effectively determined. Eq (11) and Eq (12) represent the cost function for quantum based PINNs. Also, the corresponding value of M_u is

$$F_{\psi} = 1 - |\langle\psi|\varphi\rangle|^2 \tag{13}$$

Minimizing $F' = F_R + F_I + F_{\psi}$ outputs a model for a given data set. When it comes to considering discrete models, one can construct the wave function ψ as a state vector $|\psi\rangle$ in an arbitrary computational basis $\{|0\rangle, |1\rangle, \ldots, |n\rangle\}$, in which the Hamiltonian becomes represented by a $n \times n$ matrix.

4. Experimental setup

In this study, we rely on the PyQu quantum simulator and a series of Python modules to enable efficient vector calculations (on the CPU) and additional optimizers not included in TensorFlow/Keras framework. We use Python 3.10.4, NumPy (1.22.4), and SciPy (1.8.1) modules. For the sake of simplicity, to evaluate the quantum based PINNs, we use a simple one-dimensional Schrödinger equation with dirichlet boundary conditions fixed to zero, as shown in Eq (15). The method introduced in Refs. [13,14] considers quantum-based PINNs that predict both ψ and E. By minimizing a loss function to satisfy the one-dimensional Schrödinger equation PDE, the quantum based PINNs approximately calculate eigen solutions.

$$\left[-\frac{h^2}{2m}\Delta^2 + V(r)\right]\psi(r) = E\psi(\mathrm{r}) \tag{15}$$

The set boundary conditions are on the spatial part of the wave function. Separating the equation into the time-independent Schrodinger equation and the relationship for the time evolution of the wave function. Here, $\psi(r)$ is a wave function that simultaneously assigns a complex number to each point. The parameter m is the particle's mass, and $V(r)$ represents the potential that represents the particle's environment. h is the reduced Planck constant, with action units (energy multiplied by time). An extension of a quantum based PINNs to solve a one-dimensional Schrödinger equation would require using two qumodes to initially encode the x and y coordinates of the collocation points and having interferometers (instead of the simple rotation gate in the case of one qumode) in the QNNs to entangle the two qumodes. The residual network encodes the Laplacian operator instead of the one-dimensional derivative in the two-dimensional. The baseline QNNs hyper-parameters are the following: we set the learning rate equal to 0.01 and 0.0001 for the quadratic and sinusoidal source terms cases,

Table 1
Model performance with different numbers of layers and neurons in architecture.

		Number of Neurons		
Layers		20	30	40
2	MAE	3.85	3.23	2.84
4	MAE	2.94	2.67	2.49
6	MAE	1.29	**0.72**	**0.59**

respectively. Optimizers' performance highly depends on the initial learning rate. We completed a grid search for setting the learning rate for SGD, as it used a fixed learning rate during the training and was more susceptible to the exploding and vanishing gradient problem. The collocation points are drawn randomly within the PDEs domain at each iteration. The number of collocation points per is equal to the batch size. As baseline runs, we choose 32 for the batch size. For the boundary conditions, we also evaluate the cost functions at the boundaries with several collocation points equal to batch size. The quantum circuit parameters are initialized with a normal distribution centered at zero and a standard deviation of 0.05. The simulation of a quantum based PINNs implemented with PyQu. After the training, we check the loss function values and the final error norm to characterize the accuracy of the quantum based PINNs. To evaluate the final error, we compare the analytical solution (analytical solver) and quantum-based PINNs results using a uniform grid of 32 points (including the boundary points) and take the Euclidean norm of the quantum-based PINNs approximated solution minus the analytical solution on the 32 points.

4.1. Results

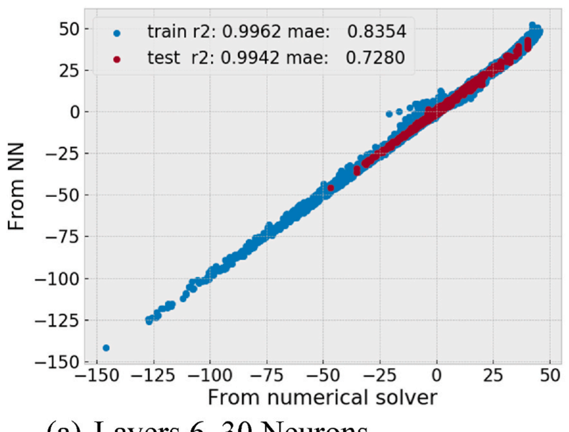
Quantum simulation generates data containing outliers. Therefore, it must be robust against outliers to mitigate their harmful effects. To ensure that the quantum-based PINNs analyze only reliable data, we incorporate the data-cleaning process based on $V(x), \psi(x)$ and add a recalibration step that is executed whenever the data is assessed as either moderate or low quality. Whenever recalibration is initiated, the device noise level is adjusted, and another measurement is taken at the same point. Then solve PDEs or reconstruct solutions on the cleaned data as shown in Fig. 6(a), and train on data as shown in Fig. 6(b). We collected 10,000 training data in a consistent pattern in the domain to determine the ideal number of hidden layers and neurons. We trained the network for 2, 4, and 6 hidden layers, each with 20, 30, and 40 neurons, and compared the MSE and inverted velocity results. These findings are shown in Table 1, which demonstrates that choosing 6 layers and 40 neurons, respectively, will yield the highest performance.

The quantum-based PINNs performed moderately well with an MAE of 0.72, shown in Fig. 7(a). The model performed exceptionally well with this added training, layers, and neurons, with an MAE of 0.59, shown in Fig. 7(b). Given the vast variability in this later group of options, it is impressive that the quantum-based PINNs could learn to forecast energy with such a high degree of accuracy.

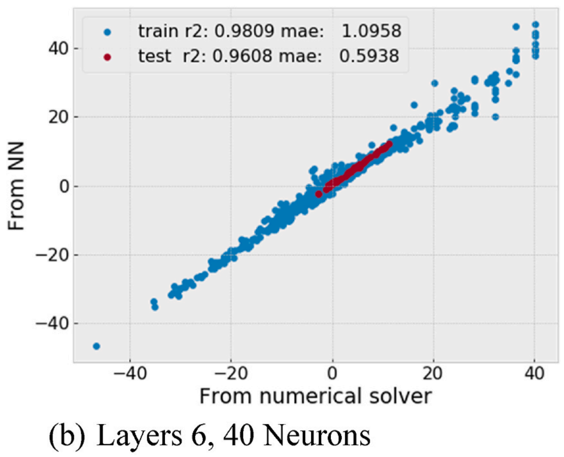
4. Conclusion

Numerous additional ML techniques, such as random forests and kernel ridge regression, have historically succeeded. Convolutional DNNs can "learn" pertinent characteristics and create a nonlinear input-to-output mapping like these techniques without formulating an input representation [71–73]. These techniques perform worse in our tests and scale, so many training instances are impractical. The efficient training and assessment of the DNNs need homogeneity in the input size, which is a significant constraint of our strategy. This paper introduced a mathematical model of quantum PINNs, a new quantum class of universal function approximators, and evaluated the mathematical model. Therefore, this review must be understood as exploratory studies that propose angles to approach the problem of using QML algorithms to understand fundamental particles. Recent research showed that PINNs is widely used in many scientific disciplines since the usage results are very positively rated. We discussed the potential of studying quantum data and the practical aspects of experimenting with QML. These difficulties placed QML in a particularly bad situation. The effectiveness of an ML algorithm is typically assessed using empirical benchmarks using fictitious datasets. With quick convergence to the correct answers, this method will enable QML processes to handle some of the most challenging data optimization issues. We have demonstrated how our proposed quantum based PINNs may be applied in QML protocols using conventional information technology in a hybrid environment.

In this work, the Schrödinger, which governs the time evolution of the quantum mechanical wave function, shows a remarkable resemblance to the core premises of proposed quantum based PINNs, and thus is natural for QC machines to work in this manner with proper Hamiltonian shaping. Big data processing methods could be a potential use case for quantum based PINNs. The capacity of a quantum-based PINNs to quickly learn how to solve a collection of PDEs approximatively has been shown in the end. A generalizable, transportable, However, would impact all theoretical physics and mathematics domains; this method can be extended to other models, the key requirement being the model's ability to be differentiated accurately and efficiently with respect to its input and parameters. Furthermore, we have identified physics-based performance metrics and control parameters necessary for enhancing performance. While modern-day quantum computers are still in the noisy intermediate-scale quantum era, it seems rational to use hybrid



(a) Layers 6, 30 Neurons



(b) Layers 6, 40 Neurons

Fig. 7. Performance of the actual vs. predicted energies for the NNs (quantum based PINNs) model and numerical solver trained.

classically inspired solutions for future endeavors.

Author contributions statement

S.R.V. and S.N.B. have contributed equally to the manuscript, S.R.V. and S.N.B. conducted the experiment(s), S.R.V. and S.N.B. analyzed the results. All authors have written and reviewed the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Steane A. Quantum computing. Rep Prog Phys 1998;61(2):117.
[2] Wojcieszyn F. Springer nature. Introduction to Quantum Computing with Q# and QDK. Springer; 2022. 1st ed, ISBN-13 978-3030993788M.
[3] Outeiral C, et al. The prospects of quantum computing in computational molecular biology. Wiley Interdisciplinary Reviews. Computational Molecular Science 2021; 11(1):e1481.
[4] Silva V. Practical quantum computing for developers: programming quantum rigs in the cloud using Python, quantum assembly language and IBM QExperience. Apress; 2018.
[5] Furia, C.A., *Quantum informatics: A survey*. Calif. Inst. Techn. Publ.–2006.–[Электронный ресурс]. URL: http://home.dei.polimi.it/furia/publs/quantuminformatics06.pdf,1-58, 2006.
[6] Li J, et al. An efficient exact quantum algorithm for the integer square-free decomposition problem. Sci Rep 2012;2(1):1–5.
[7] Schrödinger E. Quantisierung als eigenwertproblem. Ann Phys 1926;385(13): 437–90.
[8] Ballentine LE. World scientific publishing company. Quantum mechanics: a modern development 2014, 2nd Edition, 526-671.
[9] Mohseni M, et al. Commercialize quantum technologies in five years. Nature 2017; 543(7644):171–4.
[10] Zhang C, Ma Y. Ensemble machine learning: methods and applications. Springer; 2012.
[11] Vamathevan J, et al. Applications of machine learning in drug discovery and development. Nat Rev Drug Discov 2019;18(6):463–77.
[12] Recknagel F. Applications of machine learning to ecological modelling. Ecol Model 2001;146(1–3):303–10.
[13] Libbrecht MW, Noble WS. Machine learning applications in genetics and genomics. Nat Rev Genet 2015;16(6):321–32.
[14] Yao Q, et al. Taking human out of learning applications: a survey on automated machine learning. 2018. rXiv preprint arXiv:1810.13306.
[15] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, George E. Dahl *Proceedings of the 34th International Conference on Machine Learning*, PMLR 70: 1263-1272, 1-10,2017.
[16] Dral PO. Quantum chemistry in the age of machine learning. J Phys Chem Lett 2020;11(6):2336–47.
[17] Schütt KT, et al. Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions. Nat Commun 2019;10(1):1–10.
[18] Carpenter GA, Grossberg S. The ART of adaptive pattern recognition by a self-organizing neural network. Computer 1988;21(3):77–88.
[19] Fukushima K. A neural network for visual pattern recognition. Computer 1988;21 (3):65–75.
[20] Yu Q, et al. A spiking neural network system for robust sequence recognition. IEEE Transact Neural Networks Learn Syst 2015;27(3):621–35.
[21] Bengio Y. Artificial neural networks and their application to sequence recognition. 1991.
[22] Gaur P. Neural networks in data mining. Int J Electron Comput Sci Eng 2012;1(3): 1449–53.
[23] Lu H, Setiono R, Liu H. Effective data mining using neural networks. IEEE Trans Knowl Data Eng 1996;8(6):957–61.
[24] Huang W, et al. Neural networks in finance and economics forecasting. Int J Inf Technol Decis Making 2007;6(1):113–40.
[25] Burrell PR, Folarin BO. The impact of neural networks in finance. Neural Comput Appl 1997;6(4):193–200.
[26] Garliauskas A. Neural network chaos and computational algorithms of forecast in finance. In: IEEE SMC'99 conference proceedings. 1999 IEEE international conference on systems, man, and cybernetics (cat. No. 99CH37028). IEEE; 1999.
[27] Thulasiram RK, Rahman RM, Thulasiraman P. Neural network training algorithms on parallel architectures for finance applications. In: 2003 international conference on parallel processing workshops, 2003. Proceedings. IEEE; 2003.
[28] Amato F, et al. Artificial neural networks in medical diagnosis. Elsevier; 2013. p. 47–58.
[29] Li Q, et al. Medical image classification with convolutional neural network. In: 2014 13th international conference on control automation robotics & vision (ICARCV). IEEE; 2014.
[30] Azar AT. Fast neural network learning algorithms for medical applications. Neural Comput Appl 2013;23(3):1019–34.
[31] Qayyum A, et al. Medical image retrieval using deep convolutional neural network. Neurocomputing 2017;266:8–20.
[32] Chen M, et al. Deep feature learning for medical image analysis with convolutional autoencoder neural network. IEEE Trans on Big Data 2017;7(4):750–8.
[33] Ultsch A. Self-organizing neural networks for visualisation and classification. In: Information and classification. Springer; 1993. p. 307–13.
[34] Liu W, Gopal S, Woodcock C. Spatial data mining for classification, visualisation and interpretation with artmap neural network. In: Data mining for scientific and engineering applications. Springer; 2001. p. 201–21.
[35] Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Network 1998;9(5):987–1000.
[36] Khoo Y, Lu J, Ying L. Solving parametric PDE problems with artificial neural networks. Eur J Appl Math 2021;32(3):421–35.
[37] Han J, Nica M, Stinchcombe AR. A derivative-free method for solving elliptic partial differential equations with deep neural networks. J Comput Phys 2020;419: 109672.
[38] Brink AR, Najera-Flores DA, Martinez C. The neural network collocation method for solving partial differential equations. Neural Comput Appl 2021;33(11): 5591–608.
[39] Deo M. Artificial neural networks in coastal and ocean engineering 2010: 39(4): 589-596.
[40] Zhang B, Govindaraju RS. Geomorphology-based artificial neural networks (GANNs) for estimation of direct runoff over watersheds. J Hydrol 2003;273(1–4): 18–34.
[41] Nagy A, Savona V. Variational quantum Monte Carlo method with a neural-network ansatz for open quantum systems. Phys Rev Lett 2019;122(25):250501.
[42] Stanley KO, et al. Designing neural networks through neuroevolution. Nat Mach Intell 2019;1(1):24–35.
[43] Zoller P, et al. Quantum information processing and communication. The European physical journal D-atomic, molecular. Optical Plasma Phys 2005;36(2):203–28.
[44] Biamonte J, et al. Quantum machine learning. Nature 2017;549(7671):195–202.
[45] Schuld M, Sinayskiy I, Petruccione F. An introduction to quantum machine learning. Contemp Phys 2015;56(2):172–85.
[46] Cerezo M, et al. Challenges and opportunities in quantum machine learning. Nat Computational Sci 2022;2(9):567–76.
[47] Martín-Guerrero JD, Lamata L. Quantum machine learning: a tutorial. Neurocomputing 2022;470:457–61.
[48] Caro MC, et al. Generalization in quantum machine learning from few training data. arXiv preprint arXiv 2021;2111. 05292.
[49] Caro MC, et al. Out-of-distribution generalization for learning quantum dynamics. arXiv preprint arXiv 2022;2204:10268.
[50] Chuang IL, Nielsen MA. Prescription for experimental determination of the dynamics of a quantum black box. J Mod Opt 1997;44(11–12):2455–67.
[51] Raissi M, Perdikaris P, Karniadakis GE. 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.
[52] Cai S, et al. Physics-informed neural networks (PINNs) for fluid mechanics: a review. Acta Mech Sin 2022:1–12.
[53] Mao Z, Jagtap AD, Karniadakis GE. Physics-informed neural networks for high-speed flows. Comput Methods Appl Mech Eng 2020;360:112789.
[54] Sahli Costabal F, et al. Physics-informed neural networks for cardiac activation mapping. Frontiers in Physics 2020;8:42.
[55] Kharazmi E, Zhang Z, Karniadakis GE. Variational physics-informed neural networks for solving partial differential equations. arXiv preprint arXiv 2019;1912. 00873.
[56] Misyris GS, Venzke A, Chatzivasileiadis S. Physics-informed neural networks for power systems. In: 2020 IEEE power & energy society general meeting (PESGM). IEEE; 2020.
[57] Xu YLi. J.Chen X.Physics informed neural networks for velocity inversionSEG. https://doi.org/10.1190/segam2019-3216823.1[OnePetro].
[58] van Herten RL, et al. Physics-informed neural networks for myocardial perfusion MRI quantification. arXiv preprint arXiv 2011;12844(2020).
[59] Vadyala SR, Betgeri SN, Betgeri NP. Physics-informed neural network method for solving one-dimensional advection equation using PyTorch. Array 2022;13: 100110.
[60] Goldsborough P. A tour of tensorflow. URL: https://doi.org/10.48550/arXiv.1610.01178.arXiv preprint arXiv:1610.01178 2016.
[61] Paszke A., et al. Automatic differentiation in pytorch. NIPS 2017 Workshop Autodiff Decision Program Chairs:1-4,2017.
[62] Kyriienko O, Paine AE, Elfving VE. Solving nonlinear differential equations with differentiable quantum circuits. Phys Rev 2021;103(5):52416.
[63] Bai Y, Chaolu T, Bilige S. Physics informed by deep learning: numerical solutions of modified Korteweg-de Vries equation. Adv Mathematical Phys 2021;2021.
[64] Li C. A partial differential equation-based image restoration Method in environmental art design. Adv Mathematical Phys 2021;2021.
[65] Ulbrich M, Ulbrich S. *Automatic differentiation: a structure-exploiting forward mode with almost optimal complexity for Kantorovič trees*, in *Applied mathematics and parallel computing*. Springer; 1996. p. 327–57.

[66] Vadyala SR, et al. A review of physics-based machine learning in civil engineering. Res Eng 2021:100316.
[67] Mattheakis M, et al. First principles physics-informed neural network for quantum wavefunctions and eigenvalue surfaces. 2022. URL: https://doi.org/10.48550/arXiv.2211.04607.arXivpreprintarXiv:2211.04607.
[68] Ziman M, et al. Diluting quantum information: An analysis of information transfer in system-reservoir interactions. Phys Rev 2002;65(4):42105.
[69] Kjaergaard M, et al. Superconducting qubits: Current state of play. Annual Review of Condensed Matter Physics 2020;11:369–95.
[70] Bruß D, Macchiavello C. Approximate quantum cloning. Quantum Information: From Foundations to Quantum Technology Applications 2016:55–74.
[71] Vovk V. *Kernel ridge regression*, in *Empirical inference*. Springer; 2013. p. 105–16.
[72] Breiman L. Random forests. Mach Learn 2001;45(1):5–32.
[73] Lucas A, et al. Using deep neural networks for inverse problems in imaging: beyond analytical methods. IEEE Signal Process Mag 2018;35(1):20–36.