

# Quantum physics-informed neural network with residual-based adaptive refinement for solving partial differential equations

Le Li,<sup>1</sup> Junkai Yang,<sup>1</sup> Qingle Wang,<sup>2</sup> and Zhichao Zhang<sup>1,\*</sup>

<sup>1</sup>*School of Mathematics and Physics, University of Science and Technology Beijing, Beijing 100083, China*

<sup>2</sup>*School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China*



(Received 23 April 2026; accepted 22 June 2026; published 10 July 2026)

In this paper, we propose a quantum physics-informed neural network (QPINN) to solve partial differential equations (PDEs) by using residual-based adaptive refinement (RAR). The method incorporates a residual-driven adaptive sampling mechanism into the training of QPINN, which aims to enhance convergence and accuracy by enforcing physical constraints in regions with high residuals. Then, we conduct numerical experiments on the one-dimensional viscous Burgers' equation and diffusion equation, which have different solution structures, as well as the three-dimensional heat equation. The results show that QPINN with RAR reduces the number of parameters by about 44%, and achieves reductions of 31.65%–39.64%, 15.77%–20%, and 3.31%–23.40% over physics-informed neural network (PINN), PINN with RAR, and QPINN in mean relative  $L_2$  error, respectively. In summary, it improves model convergence and solution accuracy even with limited sampling points and fewer parameters. These advantages further illustrate that QPINN with RAR is an efficient and stable pathway to solve complex and high-dimensional PDEs.

DOI: [10.1103/lb19-wv6f](https://doi.org/10.1103/lb19-wv6f)

## I. INTRODUCTION

Partial differential equations (PDEs) play a fundamental role in a wide range of physical phenomena, including fluid dynamics [1], quantum mechanics [2], and electromagnetic propagation [3]. Accurate and efficient numerical solutions of PDEs are therefore of great importance in science and engineering. However, traditional numerical methods such as the finite difference method (FDM) and finite element method (FEM) [4] exhibit several inherent limitations. To be specific, FDM typically relies on structured grids, which makes it difficult to accurately represent irregular geometries and complex boundary conditions [5]. And FEM often requires the assembly and solution of large sparse linear systems, resulting in substantial computational cost and memory consumption for fine discretizations [6]. Furthermore, both methods suffer from the curse of dimensionality, significantly limiting their scalability in high-dimensional applications [7]. These challenges have motivated the exploration of alternative frameworks that can overcome the geometric and computational constraints of traditional methods.

In recent years, machine learning has achieved remarkable success in diverse applications [8–10]. Benefiting from its powerful nonlinear fitting and feature learning abilities, machine learning offers a novel paradigm for PDE solutions

[11–13]. Among these machine learning techniques [14,15], physics-informed neural network (PINN) [16] has quickly become a hot research topic at the intersection of machine learning and scientific computing, owing to their distinct advantages including mesh-free computation and the direct embedding of physical conservation laws as regularization constraints.

For the PINN, many researchers have conducted extensive studies on algorithm improvements and practical applications, achieving rich results. In terms of algorithmic improvements, researchers have improved the convergence speed and computational accuracy of PINN in solving complex PDEs by optimizing network structures [17], improving loss function designs [18,19], introducing adaptive activation function [20], and adaptive sampling strategies [21,22]. For the application field, PINN has been successfully applied to multiple engineering fields, including fluid mechanics [23], solid mechanics [24], electromagnetism [25], and weather prediction [26].

Meanwhile, quantum computing is a groundbreaking computing technology that has unique quantum features such as superposition and entanglement [27]. These features give it unmatched advantages when handling high-dimensional data and complex optimization problems. Currently, many quantum algorithms have become well established, such as Shor's algorithm [28], the variational quantum linear solve algorithm [29,30], quantum annealing algorithms [31], the Harrow-Hassidim-Lloyd (HHL) algorithm [32,33], and quantum machine learning (QML) [34]. Specifically, QML is the intersection of quantum computing and machine learning. Recently, some researchers have combined the computing power of quantum computing with the physical constraints of PINN. First, Xiao *et al.* [35] proposed physics-informed

\*Contact author: zhichao@ustb.edu.cn

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

quantum neural network that uses parametrized quantum circuit (PQC) [36] as function approximator to solve PDE forward and inverse problems, outperforming classical PINN on trigonometric-related equations with higher accuracy and fewer parameters. Then, Farea *et al.* [37] proposed quantum-classical physics-informed neural network, which combines quantum and classical components for solving PDEs with high parameter efficiency. Meanwhile, Berger *et al.* [38] proposed trainable embedding quantum physics-informed neural network, which employs trainable feedforward neural network embedding to enhance expressivity and efficiency for solving nonlinear PDEs on noisy intermediate-scale quantum (NISQ) devices. Furthermore, Setty *et al.* [39] proposed a self-adaptive physics-informed quantum machine learning framework, which integrates Chebyshev feature maps and self-adaptive loss weights to solve diverse differential equations with enhanced accuracy. In terms of applications, Leong *et al.* [40] proposed hybrid quantum physics-informed neural network for modeling high-speed flows. These studies indicate that the combination of QML and PINN provides an approach to address the problems that PINN encounters when solving complex and high-dimensional PDEs, such as insufficient computational efficiency and limited accuracy.

However, current quantum methods still face difficulties for accuracy and efficiency in practical applications. In general, these methods adopt fixed sampling strategies [21] such as uniform sampling, Latin hypercube sampling, Halton sequences, or Sobol sequences, which may lead to insufficient sampling in regions that are difficult to fit or oversampling in smooth regions, thereby compromising the accuracy of results. In contrast, adaptive sampling enables dynamic sampling and better balances the distribution of sampling points. For instance, Nabian *et al.* [41] proposed an importance sampling approach that prioritizes points according to the loss function. Building on this idea, Lu *et al.* [42] introduced the residual-based adaptive refinement (RAR) method, which adds new residual points in regions with high PDE residuals to improve the accuracy of the solution.

While these adaptive sampling strategies have significantly enhanced the performance of the PINN, their integration into quantum physics-informed neural network (QPINN) remains largely unexplored. Therefore, a natural idea is to apply the RAR sampling mechanism within the QPINN framework, which helps the network focus on areas with high residuals. To evaluate its effectiveness, we conduct numerical experiments on three benchmark PDEs, including the one-dimensional viscous Burgers' equation, the diffusion equation, and the three-dimensional (3D) heat equation. The results demonstrate that the proposed method outperforms PINN, QPINN, and PINN-RAR across the considered test cases. Specifically, for Burgers' equation with a nontrigonometric solution, QPINN-RAR achieves a clear improvement over the other methods. The mean relative  $L_2$  error is reduced by 37.22% compared to PINN, by 20% compared to PINN-RAR, and by 3.31% compared to QPINN. For the diffusion equation with trigonometric-function solution, the proposed method achieves 31.65%, 15.77%, and 23.40% mean error reductions relative to the PINN, PINN with RAR, and QPINN, respectively. Furthermore, for 3D heat equa-

---



---

#### ALGORITHM 1. RAR sample for QPINN.

---



---

- 1: The initial set ( $\mathcal{D}_0$ ) of training points is randomly sampled according to a continuous uniform distribution over the PDE domain.
  - 2: Train the PINN on  $\mathcal{D}_0$  for a fixed number of Epochs.
  - 3: **while** not meet termination condition **do**
  - 4:   Calculate absolute residuals on a candidate set  $\mathcal{C}$ :  
 $s(\mathbf{x}) = |r(\mathbf{x})|$ ,  $\mathbf{x} \in \mathcal{C}$ , where  $r(\mathbf{x})$  is the PDE residual.
  - 5:   Split  $\mathcal{C}$  into two groups according to parameter  $\lambda$ :  $\mathcal{P} := \{\mathbf{x} \in \mathcal{C} : \mathbf{x} \text{ is in the positive/high-residual region}\}$ ,  
 $\mathcal{N} := \mathcal{C} \setminus \mathcal{P}$   
     (by sorting  $s(\mathbf{x})$  and taking the top- $\lambda$  fraction as  $\mathcal{P}$ ).
  - 6:   Perform RAR sampling: select  $n$  points from  $\mathcal{P}$  with the largest residuals  $\mathcal{P}_{\text{sel}} := \text{Top}_n(s(\mathbf{x}) \text{ over } \mathbf{x} \in \mathcal{P})$ .
  - 7:   Form the new added set:  $\mathcal{D} := \mathcal{D}_0 \cup \mathcal{P}_{\text{sel}}$ . Update training set and continue to train.
  - 8: **end while**
- 
- 

tion, the mean error of QPINN with RAR drops by 39.64%, 16.75%, and 16.75% compared to PINN, PINN with RAR, and QPINN, respectively, and the trainable parameters of QPINN with RAR are about 44% fewer than the other two methods of PINN for the three equations. These results demonstrate that the proposed method exhibits superior performance in terms of prediction accuracy, generalization ability, and trainable parameters, thus verifying its effectiveness in solving PDEs within the hybrid quantum-classical framework.

The paper is organized as follows. In Sec. II, we present the algorithm of RAR and QPINN with RAR. Then, we demonstrate the effectiveness of QPINN with RAR for three different problems in Sec. III. Specifically, we systematically compare the performance of PINN, PINN with RAR, and QPINN with RAR. Finally, we conclude the paper in Sec. IV.

## II. METHODS

In this section, we introduce the fundamentals and framework of QPINN integrated with RAR. First, we elaborate on the implementation process of RAR. Then, we present the overall workflow of QPINN with RAR.

### A. Residual-based adaptive refinement

In the following, we introduce the process of RAR. First, we use a set of initial sampling points that are usually uniformly sampled to train the PINN for a certain number of rounds. Then, after training pauses, the residual values of the physical equation are evaluated at a large number of candidate points in the solution domain, so as to identify high-error regions with large residuals. Next, more points are added near these regions and training is continued with this new set of sampling points. Finally, this process is repeated until the sampling points gather in complex and hard-to-fit regions of the solution [43].

To understand the procedure of RAR more clearly, the detailed steps are presented in Algorithm 1, and further illustration is available in Fig. 1.

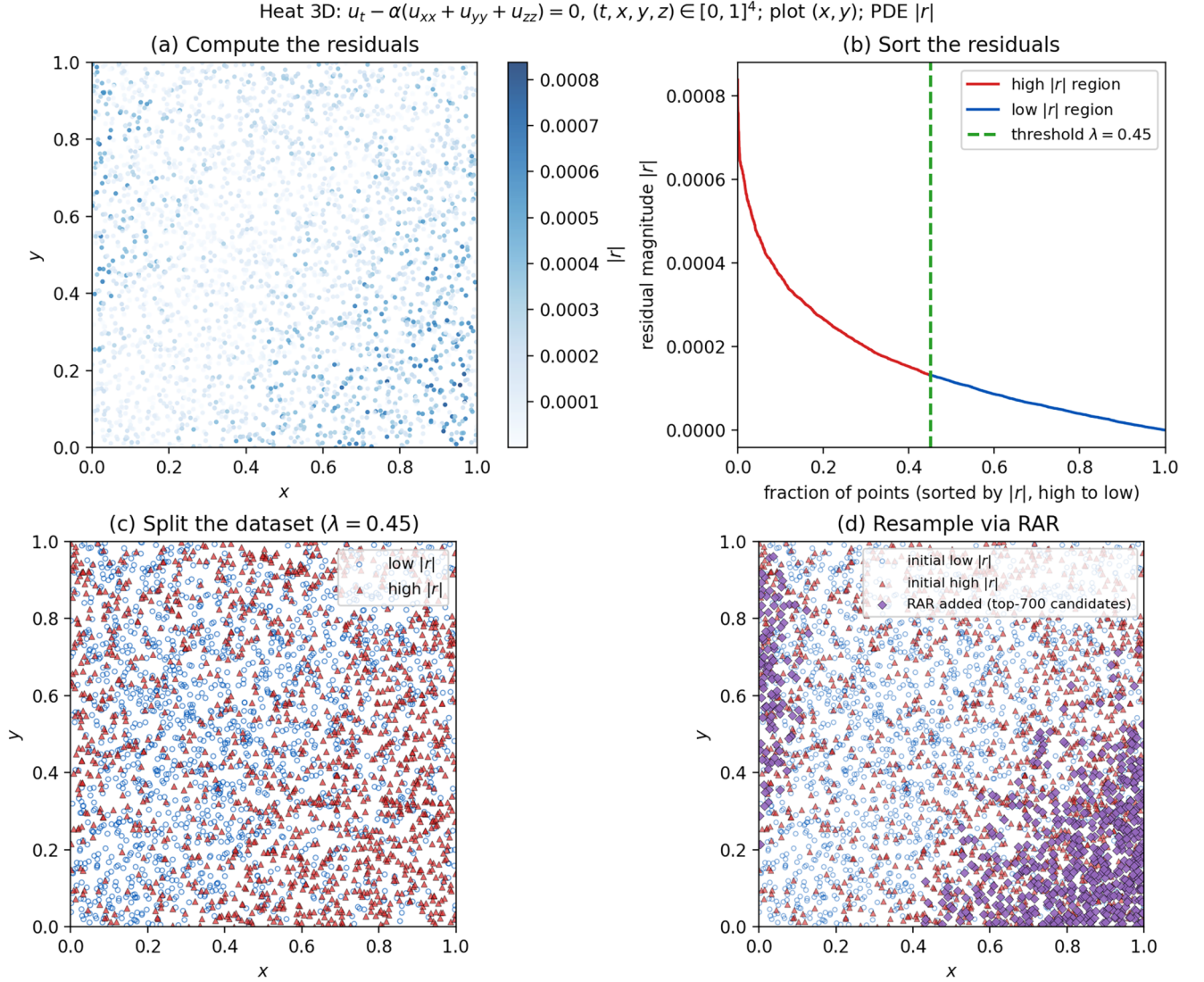


FIG. 1. RAR sampling for 3D heat equation. (a) Compute the absolute residuals. (b) Sort the residuals in descending order and use a ratio  $\lambda$  to split the dataset into two parts. (c) The distribution of sampling points divided into two parts. (d) Apply RAR to perform resampling and create a new training dataset.

### B. Quantum physics-informed neural network with RAR

In this section, we introduce the QPINN with RAR, whose detailed procedure is shown in Fig. 2. In short, the core idea of QPINN is to incorporate the governing equations into the training process of neural networks and PQCs, such that the solution of PDEs can be approximated by minimizing a physics-informed loss function. Specifically, the classical input variables are encoded into a high-dimensional Hilbert space through a PQC, generating highly nonlinear quantum feature representations. The neural network is then trained within this quantum state space, where the governing equations, boundary conditions, and initial conditions are enforced through the loss function.

To elaborate on the implementation of QPINN with RAR, we first present the general form of the nonlinear PDE as

follows:

$$\begin{aligned} \mathcal{N}[u(t, \mathbf{x})] &= f(t, \mathbf{x}), & t \in [0, T], \mathbf{x} \in \Omega, \\ u(0, \mathbf{x}) &= u_0(\mathbf{x}), & \mathbf{x} \in \Omega, t = 0, \\ \mathcal{B}[u] &= g(\mathbf{x}, t), & t \in [0, T], \mathbf{x} \in \partial\Omega. \end{aligned} \quad (1)$$

Here,  $\mathcal{N}[\cdot]$  is a linear or nonlinear differential operator,  $u(t, \mathbf{x})$  denotes the physical quantity to be solved,  $f(t, \mathbf{x})$  is the known source term in the differential equation,  $\Omega$  represents the spatial domain, and  $\mathcal{B}[\cdot]$  is the boundary operator corresponding to the boundary conditions.

Then, the residual can be expressed as

$$r(t, \mathbf{x}) = \mathcal{N}[u_\theta(t, \mathbf{x})] - f(t, \mathbf{x}),$$



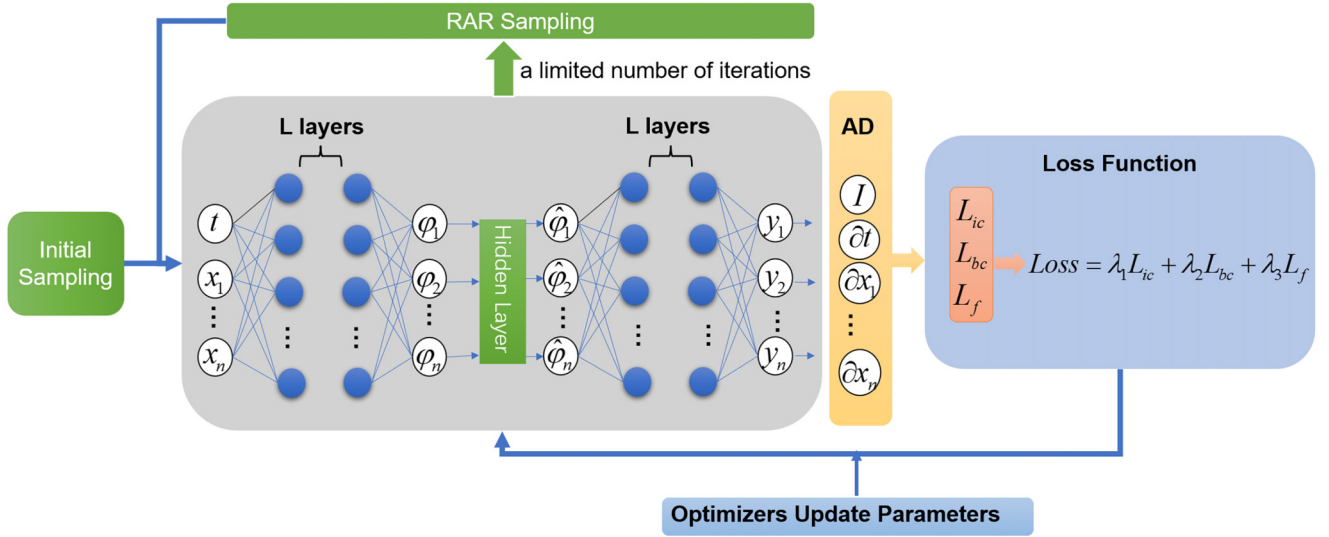


FIG. 2. Schematic of QPINN with RAR. The NNs take spatial and temporal inputs and predict the solution variables. Automatic differentiation is used to compute derivatives, which are then used to formulate the loss function incorporating PDE residuals, boundary conditions, and initial conditions. An optimizer updates the network parameters to minimize the loss and enforce the physics constraints. After the initial sampling, RAR sampling is performed following a certain number of training iterations, and then the training process is resumed.

where  $u_\theta(t, x)$  is the approximation of the solution, and the loss function of PINN consists of the following components:

Initial condition loss [44]:

$$\mathcal{L}_{ic}(\theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \|u_\theta(0, \mathbf{x}_{ic,i}) - u_0(\mathbf{x}_{ic,i})\|^2. \quad (2)$$

Boundary condition loss:

$$\mathcal{L}_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} \|\mathcal{B}[u_\theta](t_{bc,i}, \mathbf{x}_{bc,i}) - g(\mathbf{x}_{bc,i}, t_{bc,i})\|^2. \quad (3)$$

Physical term loss:

$$\mathcal{L}_f(\theta) = \frac{1}{N_f} \sum_{i=1}^{N_f} \|\mathcal{N}[u_\theta](t_{f,i}, \mathbf{x}_{f,i}) - f(t_{f,i}, \mathbf{x}_{f,i})\|^2. \quad (4)$$

Total loss:

$$\mathcal{L}(\theta) = \lambda_{ic}\mathcal{L}_{ic}(\theta) + \lambda_{bc}\mathcal{L}_{bc}(\theta) + \lambda_f\mathcal{L}_f(\theta). \quad (5)$$

Here,  $\lambda_{ic}$ ,  $\lambda_{bc}$ ,  $\lambda_f$  denote the weights of the respective loss terms.  $N_{ic}$ ,  $N_{bc}$ ,  $N_f$  represent the number of training data points for each loss term, respectively. In this work, the loss weights are  $\lambda_{ic} = 1$ ,  $\lambda_{bc} = 1$ , and  $\lambda_f = 1$ , respectively.

Besides, the preprocessing and postprocessing are similar to PINN [45]. Specifically, the preprocessing layer is constructed based on a fully connected network architecture, which consists of 2 layers with 20 neurons per layer, and the Tanh is adopted as the activation function. To ensure the symmetry of the network architecture, the postprocessing layer paired with the preprocessing layer is designed with an identical structure. Additionally, PINN uses single-layer neural network with 20 neurons in the hidden layer, while QPINN adopts a PQC [37], which is shown in Fig. 3.

The PQC adopts a hardware-efficient variational architecture with an angle embedding layer, parametrized single-qubit rotation gates (RX and RZ), and entangling gates. Classical

data are encoded into quantum states via angle embedding, followed by multiple layers of trainable rotations and local entanglement to enhance circuit expressivity. Due to its shallow depth, low gate count, and use of native quantum gates readily available on current quantum processors, this design is highly suitable for NISQ devices. Compared with traditional PINNs, it can drastically reduce the number of trainable parameters while maintaining expressive power.

### III. NUMERICAL EXAMPLES

In this section, we conduct numerical experiments on three PDEs with different solution characteristics, including the Burgers' equation of which solution is trigonometric-function structure, diffusion equation with nontrigonometric solution, and three-dimensional heat equation. All computations are performed on the open-source frameworks of PENNYLANE [46] and PYTORCH [47].

Besides, we need to specify some key parameters and the tools employed in the numerical computations. First, the initial collocation points are generated via uniform random sampling, and the model is trained using the Adam and Limited-memory Broyden Fletcher-Goldfarb-Shanno (L-BFGS) optimizers [48]. Specifically, the Adam optimizer with a learning rate of 0.001 is adopted in the first stage for 10 000 iterations. In this process, RAR sampling is performed every 2000 iterations, with  $n$  set to 100 for the number of additional collocation points sampled from high-residual regions and added to the training set at each refinement stage. The adaptive refinement procedure continues until the maximum number of iterations is reached or the mean residual evaluated over the candidate points falls below the prescribed threshold of  $5 \times 10^{-4}$ . Afterward, the optimization is switched to the L-BFGS optimizer for precision optimization until convergence. Moreover, the L-BFGS optimizer is configured with hyperparameters: `lbfgs_factor` = 1000, `lbfgs_pgtol` =  $10^{-14}$ ,

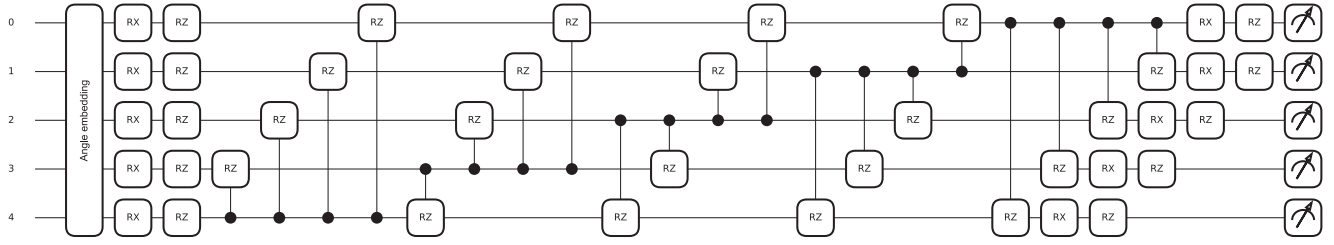


FIG. 3. Quantum circuit with five qubits and angle embedding.

the maximum iterations is 10 000, and the batch size is set to 64.

Additionally, the numbers of collocation, boundary, and initial points for all equations are selected as 500, 50, and 50, respectively. For performance evaluation, 10 000 points are randomly sampled from the remaining points that are not used for training. Next, we conduct numerical experiments on three PDEs.

#### A. One-dimensional viscous Burgers' equation

In this section, we consider the one-dimensional viscous Burgers' equation of which solution is not a trigonometric function:

$$u_t + uu_x = \nu u_{xx}. \quad (6)$$

Here, the viscosity is set as  $\nu = 0.01$ , and the computational domain is defined by  $(x, t) \in [-1, 1] \times [0, 1]$ . The exact solution subject to Dirichlet boundary conditions is given by

$$u(x, t) = \frac{\frac{x}{t+1}}{1 + \sqrt{\frac{t+1}{t_0}} e^{\frac{x^2}{4\nu(t+1)}}}. \quad (7)$$

Next, we compare the results of PINN, PINN-RAR, and QPINN-RAR, which are summarized in Table I, in which the

relative  $L_2$  error is defined as

$$\text{relative } L_2 \text{ error} = \frac{\sqrt{\sum_{i=1}^N [\hat{u}_i - u(x_i, t)]^2}}{\sqrt{\sum_{i=1}^N [u(x_i, t)]^2}},$$

where  $\hat{u}_i$  is the predicted solution and  $u(x_i, t)$  is the exact solution. In addition,  $N_p$  denotes the number of trainable parameters, the Loss refers to the final convergence loss, the Epoch is the total number of iterations at convergence, and the values are reported as the mean  $\pm$  standard deviation across ten independent runs in this paper.

As illustrated in Table I, QPINN-RAR delivers higher parameter efficiency, with the number of trainable parameters reduced by 44.27% relative to PINN and PINN-RAR. In terms of training convergence, QPINN-RAR obtains the lowest mean loss and the smallest fluctuation among the four methods, showing stronger convergence stability. For prediction accuracy, QPINN-RAR reduces the mean error by about 37.22% compared with PINN and 20% compared with PINN-RAR, which only needs about 5% more epochs. Moreover, compared with the QPINN, QPINN-RAR further lowers the mean relative  $L_2$  error by around 3.31%, while maintaining a comparable number of training epochs.

To clearly present the results of these methods, we plot the variations of loss and relative  $L_2$  error with the number of iterations, which are shown in Fig. 4. Moreover, we perform normalization on the iteration parameter represented by the

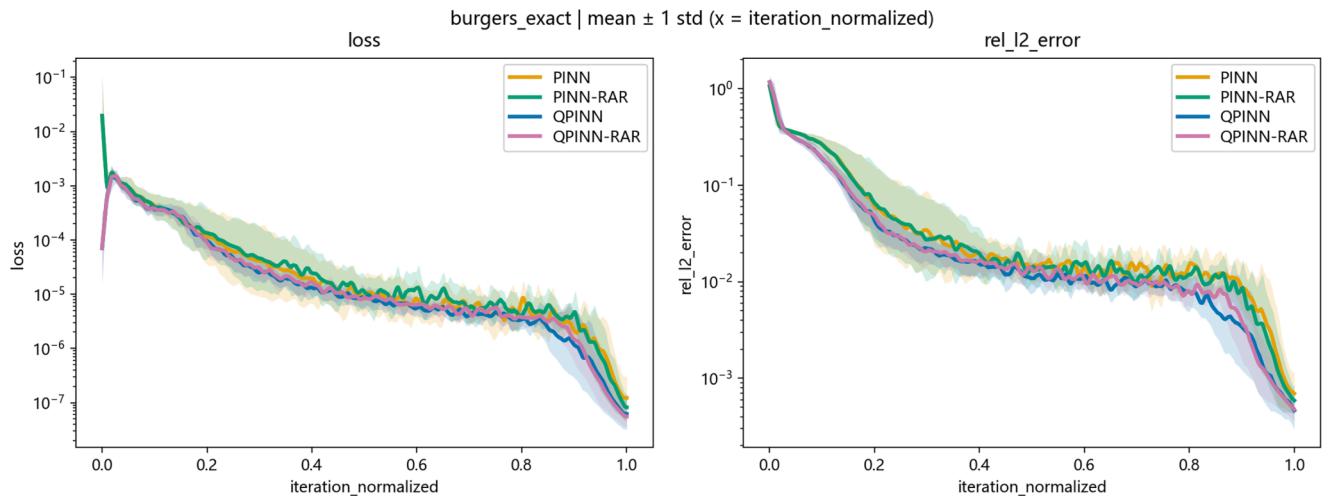


FIG. 4. Loss curves and relative  $L_2$  error curves for Burgers' equation. Pink line: mean values of the QPINN-RAR method; blue line: mean values of the QPINN method; yellow line: mean values of the PINN method; and green line: mean values of the PINN-RAR method. The shaded areas represent the corresponding standard deviations. Iteration parameter normalized for fair and intuitive comparison.

TABLE I. Comparison of four methods for Burgers' equation. Bold values denote the optimal performance.

| Method    | $N_p$       | Loss ( $\times 10^{-8}$ )         | $L_2$ ( $\times 10^{-4}$ )        | Epoch                               |
|-----------|-------------|-----------------------------------|-----------------------------------|-------------------------------------|
| PINN      | 2128        | $18.70 \pm 19.90$                 | $7.90 \pm 4.33$                   | <b><math>10\,642 \pm 341</math></b> |
| PINN-RAR  | 2128        | $9.96 \pm 7.85$                   | $6.20 \pm 2.04$                   | $10\,750 \pm 360$                   |
| QPINN     | <b>1186</b> | $7.67 \pm 5.58$                   | $5.13 \pm 2.44$                   | $11\,201 \pm 622$                   |
| QPINN-RAR | <b>1186</b> | <b><math>6.05 \pm 2.39</math></b> | <b><math>4.96 \pm 1.38</math></b> | $11\,140 \pm 360$                   |

horizontal axis to enable a fair and intuitive comparison in this paper.

From Fig. 4, we observe that the training loss and relative  $L_2$  error curves for four methods exhibit a consistently de-

creasing trend with iterations. Among them, the PINN method shows the slowest convergence, largest final error, and highest variance throughout the training process, reflecting its limited approximation and stability. Then, PINN-RAR achieves better performance than the PINN by incorporating RAR, which reduces fluctuation and accelerates convergence. In addition, QPINN-RAR achieves the fastest convergence rate, the lowest final error, and the narrowest standard deviation band, indicating superior accuracy and robustness.

Finally, Fig. 5 shows the solutions obtained by the four methods under the same random seed, along with their relative  $L_2$  error distributions. As shown in Figs. 5(a)–5(e), all methods capture the overall spatiotemporal evolution of the Burgers' equation and produce predictions visually consistent with the exact solution. However, the error maps in Figs. 5(f)–5(i) reveal clear differences in accuracy.

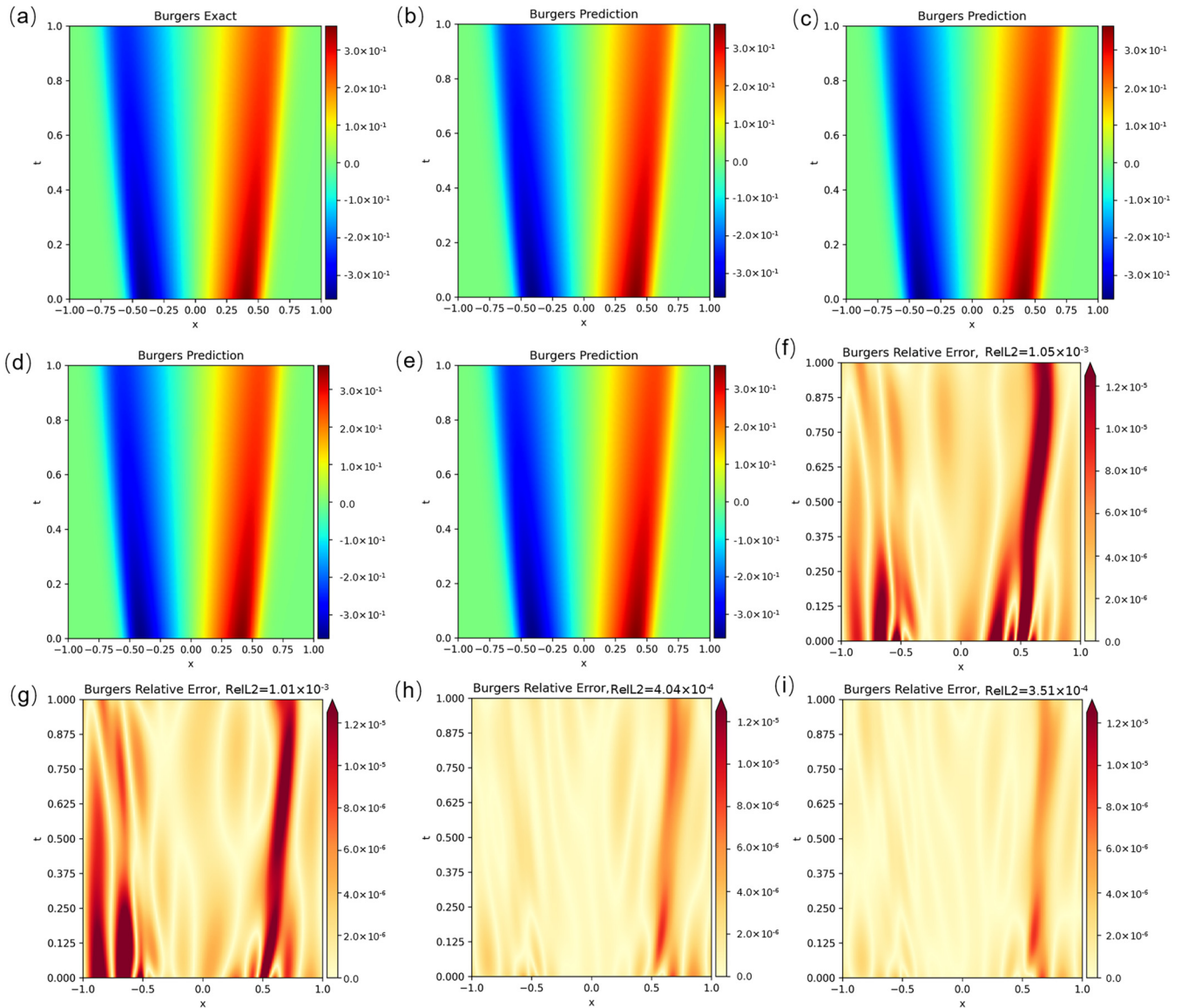


FIG. 5. Comparison of the exact solution and numerical results for the Burgers' equation. Panel (a) shows the exact solution. Panels (b)–(e) present the predicted solutions obtained by PINN, PINN with RAR, QPINN, and QPINN with RAR, respectively. Panels (f)–(i) show the corresponding relative error distributions, with the global relative  $L_2$  errors indicated in the subplot titles.

TABLE II. Comparison of four methods for diffusion equation. Bold values denote the optimal performance.

| Method    | $N_p$       | Loss ( $\times 10^{-6}$ )         | $L_2$ ( $\times 10^{-4}$ )        | Epoch                              |
|-----------|-------------|-----------------------------------|-----------------------------------|------------------------------------|
| PINN      | 2128        | $1.37 \pm 0.90$                   | $2.97 \pm 1.07$                   | $10\,793 \pm 194$                  |
| PINN-RAR  | 2128        | $1.65 \pm 2.61$                   | $2.41 \pm 1.26$                   | $10\,890 \pm 449$                  |
| QPINN     | <b>1186</b> | $1.28 \pm 1.27$                   | $2.65 \pm 1.17$                   | <b>10 596 <math>\pm</math> 381</b> |
| QPINN-RAR | <b>1186</b> | <b><math>0.80 \pm 0.66</math></b> | <b><math>2.03 \pm 0.83</math></b> | $10\,769 \pm 263$                  |

Specifically, the PINN exhibits larger errors, particularly in regions with steep gradients. Then, the PINN-RAR effectively reduces errors in these areas by introducing RAR. Among all methods, QPINN-RAR achieves the lowest and most localized errors, indicating superior approximation capability and robustness.

### B. One-dimensional diffusion equation

The above example is the type of equation of which solution is not a trigonometric function. To verify the versatility of QPINN with RAR, next we consider the following one-dimensional diffusion equation, which has trigonometric-function solution:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = v \frac{\partial^2 u}{\partial x^2}, \quad x \in [-1, 1], \quad t \in [0, T] \quad (8)$$

Among them,  $u(x, t)$  represents the concentration of the diffused substance, which can be interpreted as the probability density function of a Brownian particle at position  $x$  and time  $t$ , and the initial and boundary conditions are as follows:

$$\begin{aligned} u(x, 0) &= \sin(\pi x), \\ u(-1, t) &= u(1, t) = 0. \end{aligned}$$

The exact solution is  $u(x, t) = \sin(\pi x)e^{-t}$ .

Through numerical experiments, we obtain the results that are shown in Table II. From Table II, we can see that the

QPINN-RAR achieves the lowest mean relative  $L_2$  error, which decreases by approximately 31.65% compared to PINN and by about 15.77% compared to PINN-RAR. Moreover, it achieves the smallest mean loss value and standard deviation, demonstrating its excellent stability. In terms of training cost, all methods need similar numbers of training epochs. Furthermore, compared with QPINN, the incorporation of RAR leads to a 23.40% decrease in the mean relative  $L_2$  error. As shown in Fig. 6, QPINN-RAR consistently maintains the lowest loss and relative  $L_2$  error throughout most of the training process. Finally, similar to Burgers' equation, Fig. 7 presents the exact solution, the predicted solutions obtained by different methods, and their corresponding relative  $L_2$  error distributions. It can be observed that the PINN exhibits relatively larger local errors, particularly near the boundary regions, where error accumulation is more pronounced. Then, the PINN-RAR effectively alleviates these high-error regions and improves the solution accuracy by incorporating the RAR strategy. Among all methods, QPINN-RAR achieves the most uniform error distribution and the smallest relative  $L_2$  error.

### C. Three-dimensional heat equation

In the previous two examples, we introduced two types of one-dimensional equations, and QPINN-RAR achieved favorable performance on both. Next, we discuss the higher-dimensional case by considering the three-dimensional heat equation.

$$\frac{\partial u}{\partial t} = 0.1 \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right),$$

$$u(0, y, z, t) = u(1, y, z, t) = 0,$$

$$u(x, 0, z, t) = u(x, 1, z, t) = 0,$$

$$u(x, y, 0, t) = u(x, y, 1, t) = 0,$$

$$u(x, y, z, 0) = \sin(\pi x) \sin(\pi y) \sin(\pi z). \quad (9)$$

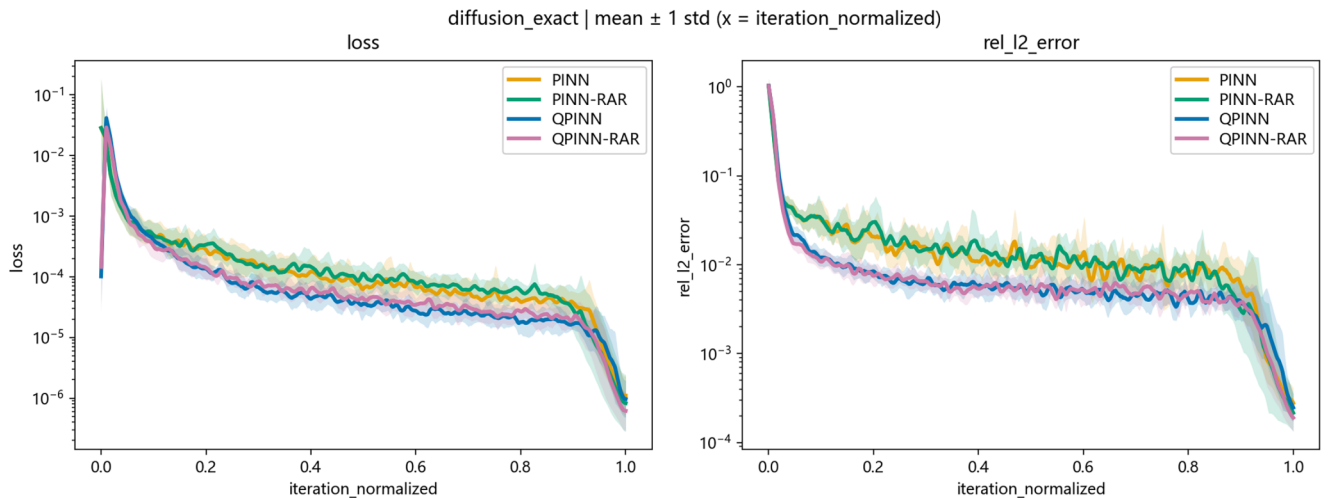


FIG. 6. Loss curves and relative  $L_2$  error curves for the diffusion equation. Pink line: mean values of the QPINN-RAR method; blue line: mean values of the QPINN method; yellow line: mean values of PINN method; and green line: mean values of PINN-RAR method. The shaded areas represent the corresponding standard deviations. Iteration parameter normalized for fair and intuitive comparison.



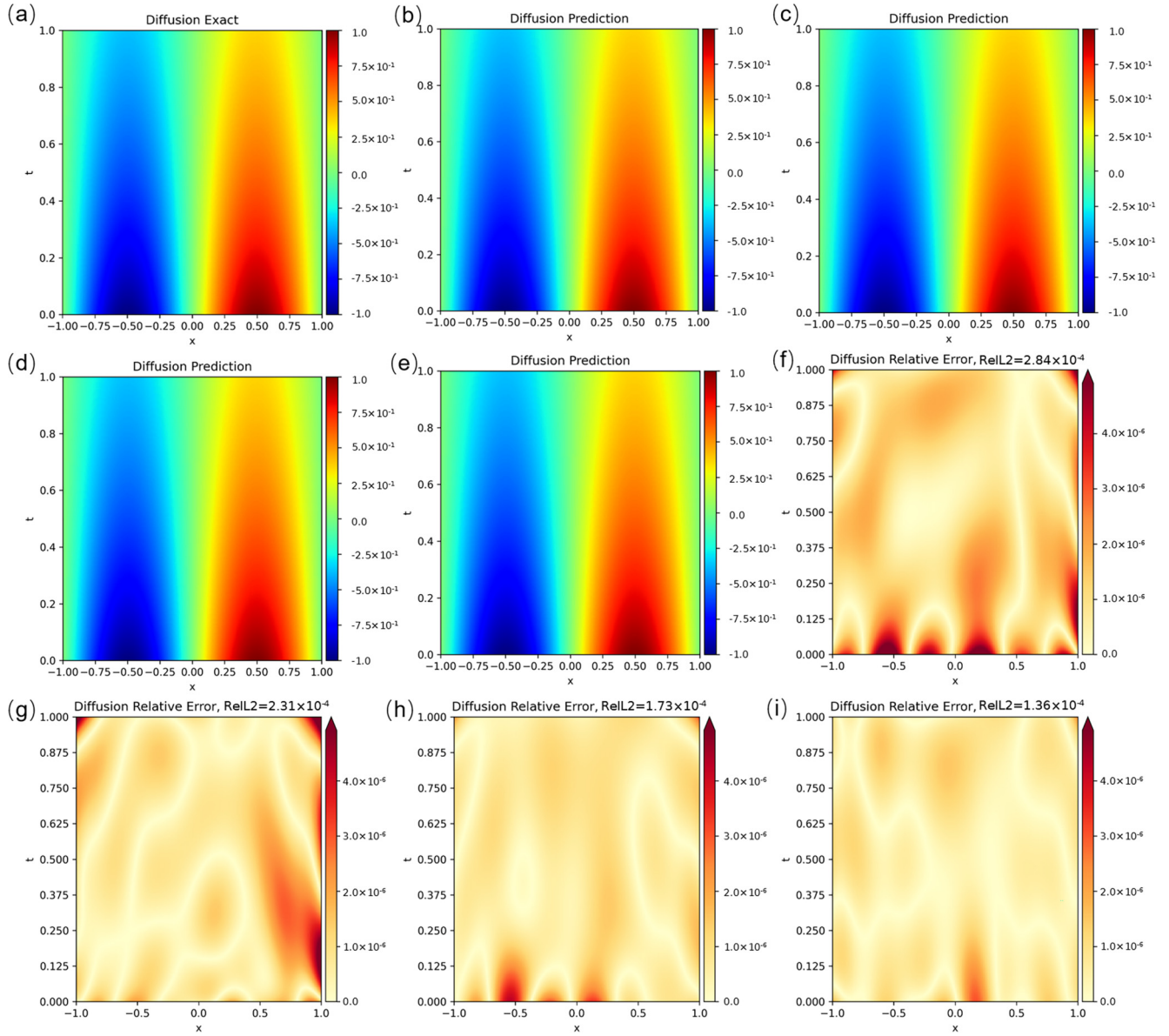


FIG. 7. Comparison of the exact solution and numerical results for the diffusion equation. Panel (a) shows the exact solution. Panels (b)–(e) present the predicted solutions obtained by PINN, PINN with RAR, QPINN, and QPINN with RAR, respectively. Panels (f)–(i) show the corresponding relative error distributions, with the global relative  $L_2$  errors indicated in the subplot titles.

The computational domain is defined by  $(x, y, z) \in [0, 1]^3$ ,  $t \in (0, 2)$ , and the exact solution is given by  $u_{\text{exact}}(x, y, z, t) = e^{-0.3\pi^2 t} \sin(\pi x) \sin(\pi y) \sin(\pi z)$ .

As can be observed from Table III, we conduct a comprehensive comparison among PINN, PINN-RAR, QPINN, and QPINN-RAR for solving the 3D heat equation. In terms of prediction accuracy, the mean  $L_2$  error of QPINN-RAR is reduced by around 39.64% relative to the PINN and approximately 16.75% compared with PINN-RAR, and it also achieves an obvious accuracy improvement over the QPINN, while benefiting from the quantum network structure, both QPINN and QPINN-RAR adopt substantially fewer trainable

parameters, which is 44.80% less than the quantity used by PINN and PINN-RAR, and QPINN-RAR decreases the mean

TABLE III. Comparison of four methods for 3D heat equation. Bold values denote the optimal performance.

| Method    | $N_p$       | Loss ( $\times 10^{-6}$ )         | $L_2$ ( $\times 10^{-2}$ )        | Epoch                               |
|-----------|-------------|-----------------------------------|-----------------------------------|-------------------------------------|
| PINN      | 2221        | $2.72 \pm 1.31$                   | $2.80 \pm 1.09$                   | $16\,020 \pm 1028$                  |
| PINN-RAR  | 2221        | <b><math>2.58 \pm 1.75</math></b> | $2.03 \pm 0.54$                   | <b><math>15\,915 \pm 623</math></b> |
| QPINN     | <b>1226</b> | $9.48 \pm 5.37$                   | $2.03 \pm 0.57$                   | $16\,081 \pm 1295$                  |
| QPINN-RAR | <b>1226</b> | $5.44 \pm 2.92$                   | <b><math>1.69 \pm 0.62</math></b> | $16\,924 \pm 843$                   |



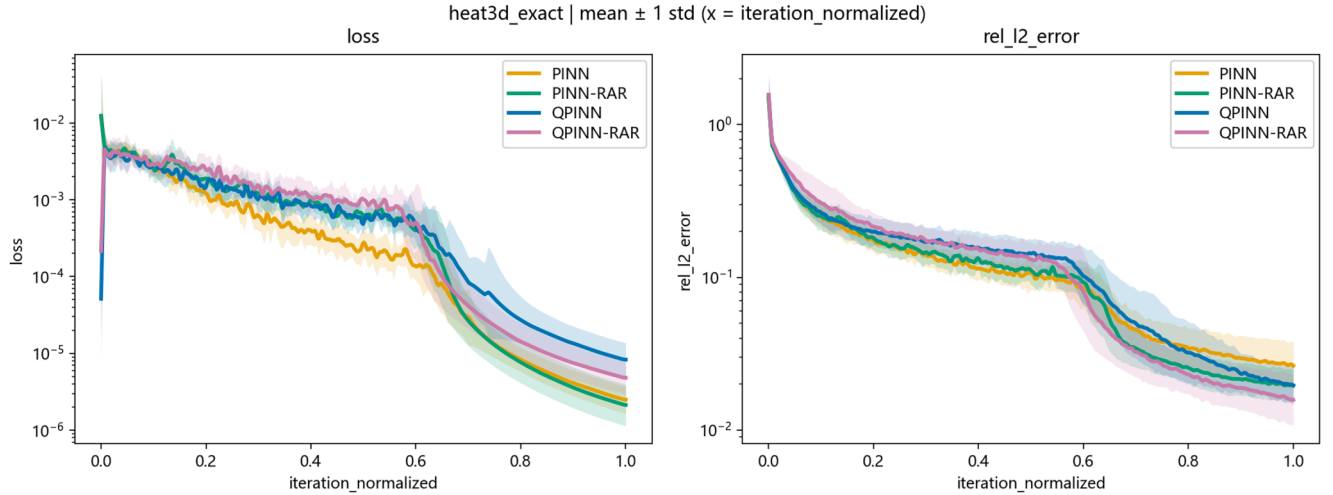


FIG. 8. Loss curves and relative  $L_2$  error curves for the 3D heat equation. Pink line: mean values of the QPINN-RAR method; blue line: mean values of the QPINN method; yellow line: mean values of the PINN method; and green line: mean values of the PINN-RAR method. The shaded areas represent the corresponding standard deviations. Iteration parameter normalized for fair and intuitive comparison.

relative  $L_2$  error by approximately 16.75% compared with QPINN, demonstrating the effectiveness of adaptive sampling in improving solution accuracy. Then, Fig. 8 clearly demonstrates the performance differences among QPINN-RAR, QPINN, PINN, and PINN-RAR methods for solving the 3D heat equation. Among them, the PINN shows the slowest convergence and the highest final error. Then, the PINN-RAR achieves the lowest final loss value and exhibits good training stability, significantly outperforming the PINN in terms of accuracy. Meanwhile, in the later stages of training, QPINN-RAR attains the lowest final relative  $L_2$  error, demonstrating superior convergence and the best overall predictive accuracy.

Similarly, Fig. 9 compares the exact solution, predicted solutions, and corresponding relative error distributions for the 3D heat equation. Both PINN-RAR and QPINN reduce prediction errors compared to PINN, demonstrating the effectiveness of RAR and quantum feature mapping, respectively. Importantly, QPINN-RAR achieves the smallest relative  $L_2$  error and the most uniform error distribution, indicating that the combination of QPINN and RAR maximizes the network's expressiveness and efficiently allocates training points to high-residual regions.

#### IV. CONCLUSION

In this work, we integrate the RAR strategy into the QPINN framework to solve PDEs. By dynamically placing collocation points in high-residual regions, the proposed QPINN-RAR method better enforces physical constraints and boosts training efficiency, and extensive numerical experiments on PDEs with diverse solution characteristics show that QPINN-RAR achieves higher prediction accuracy, more stable convergence,

and better parameter efficiency than traditional PINN, PINN-RAR, and QPINN. The results prove that combining quantum feature representations with adaptive sampling is a promising way to improve physics-informed learning, especially for high-dimensional PDEs and problems with complex solution structures.

While the current study is limited to scenarios where training and test data obey the same distribution, the proposed QPINN-RAR method possesses great potential for extended applicability. It is promising to generalize this method to out-of-distribution scenarios and high-dimensional problems where analytical exact solutions are unattainable, such as practical PDE simulations with variable boundary and initial conditions, as well as complex PDE problems involving turbulence, shock waves, discontinuities, and singularities. Furthermore, it is also interesting and necessary to investigate the robustness of the proposed method against practical imperfections of quantum hardware, such as noise, decoherence, and gate errors, so as to verify its stability under real quantum hardware conditions.

#### ACKNOWLEDGMENTS

This work was supported by the Beijing Natural Science Foundation (Grant No. 4252006) and the NSFC (Grant No. 62471180).

#### DATA AVAILABILITY

The data that support the findings of this article are not publicly available. The data are available from the authors upon reasonable request.

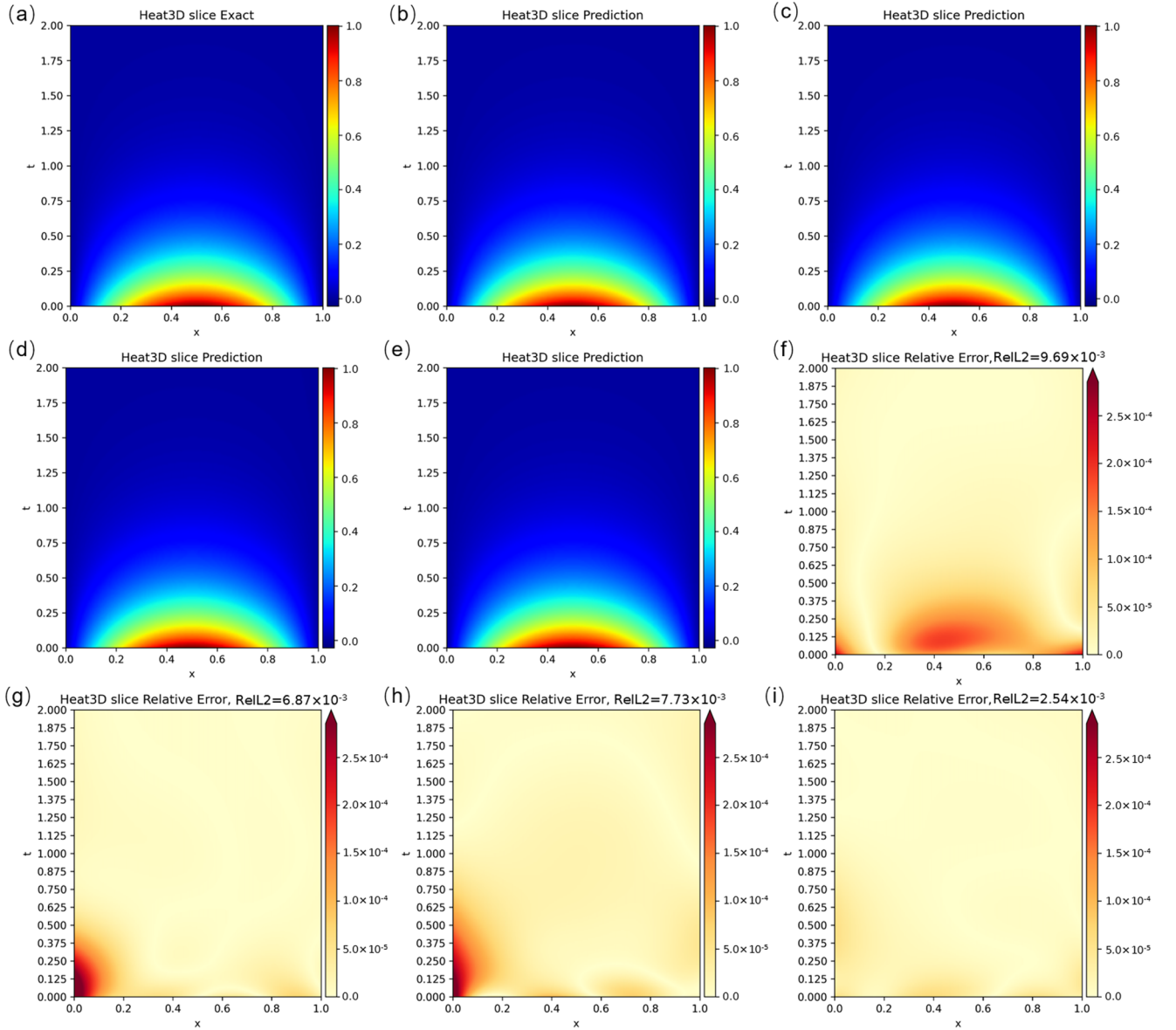


FIG. 9. Comparison of the exact solution and numerical results for the 3D heat equation. Panel (a) shows the exact solution. Panels (b)–(e) present the predicted solutions obtained by PINN, PINN with RAR, QPINN, and QPINN with RAR, respectively. Panels (f)–(i) show the corresponding relative error distributions on the 2D slice at  $y = z = 0.5$ , with the global relative  $L_2$  errors indicated in the subplot titles.

- [1] Y. Swapna, V. Nirmale, N. Pavani, C. Singh, R. Ranganathan, A. R. Thontla, and N. M. Kumar, Applications of partial differential equations in fluid physics, *Commun. Appl. Nonlinear Anal.* **31**, 207 (2024).
- [2] R. Howe, Quantum mechanics and partial differential equations, *J. Funct. Anal.* **38**, 188 (1980).
- [3] L. E. Jalil, N. H. Malik, and M. K. Hussein, A comparative study of partial differential equation solving methods and their applications, *Bilangan* **3**, 01 (2025).
- [4] A. Tezuka, Finite element and finite difference methods, in *Springer Handbook of Materials Measurement Methods*, edited by H. Czichos, T. Saito, and L. Smith, (Springer, Berlin, Heidelberg, 2006), pp. 1033–1060.
- [5] Y. Ren, H. Feng, and S. Zhao, A FFT accelerated high order finite difference method for elliptic boundary value problems over irregular domains, *J. Comput. Phys.* **448**, 110762 (2022).
- [6] S. Koric, Q. Lu, and E. Guleryuz, Evaluation of massively parallel linear sparse solvers on unstructured finite element meshes, *Comput. Struct.* **141**, 19 (2014).
- [7] J. Han, A. Jentzen, and W. E, Solving high-dimensional partial differential equations using deep learning, *Proc. Natl. Acad. Sci. USA* **115**, 8505 (2018).
- [8] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine

- learning and the physical sciences, *Rev. Mod. Phys.* **91**, 045002 (2019).
- [9] M. I. Jordan and T. M. Mitchell, Machine learning: Trends, perspectives, and prospects, *Science* **349**, 255 (2015).
  - [10] A. Cascarano, J. Mur-Petit, J. Hernández-González *et al.*, Machine and deep learning for longitudinal biomedical data: a review of methods and applications, *Artif. Intell. Rev.* **56**, 1711 (2023).
  - [11] Z. Long, Y. Lu, X. Ma, and B. Dong, PDE-Net: Learning PDEs from data, in *35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80 (PMLR, 2018), pp. 3208–3216.
  - [12] L. Mouton, F. Reiter, Y. Chen, and P. Rebentrost, Deep-learning-based quantum algorithms for solving nonlinear partial differential equations, *Phys. Rev. A* **110**, 022612 (2024).
  - [13] C. A. Amaral, V. L. Oliveira, J. P. Salazar, and E. I. Duzzioni, A review of quantum machine learning and quantum-inspired applied methods to computational fluid dynamics, *Braz. J. Phys.* **56**, 39 (2026).
  - [14] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. Leung, A survey on security threats and defensive techniques of machine learning: A data driven view, *IEEE* **6**, 12103 (2018).
  - [15] J. C. Cheng, W. Chen, K. Chen, and Q. Wang, Data-driven predictive maintenance planning framework for MEP components based on BIM and IoT using machine learning algorithms, *Autom. Constr.* **112**, 103087 (2020).
  - [16] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* **378**, 686 (2019).
  - [17] H. Jeong, J. Bai, C. P. Batuwatta-Gamage, C. Rathnayaka, Y. Zhou, and Y. Gu, A physics-informed neural network-based topology optimization (PINNTO) framework for structural optimization, *Eng. Struct.* **278**, 115484 (2023).
  - [18] J. Hua, Y. Li, C. Liu, P. Wan, and X. Liu, Physics-informed neural networks with weighted losses by uncertainty evaluation for accurate and stable prediction of manufacturing systems, *IEEE Trans. Neural Networks Learn. Syst.* **35**, 11064 (2023).
  - [19] S. Plankovskyy, Y. Tsegelnyk, N. Shyshko, I. Litvinchev, T. Romanova, and J. M. V. Cantú, Review of physics-informed neural networks: Challenges in loss function design and geometric integration, *Mathematics* **13**, 3289 (2025).
  - [20] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, *J. Comput. Phys.* **404**, 109136 (2020).
  - [21] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, *Comput. Methods Appl. Mech. Eng.* **403**, 115671 (2023).
  - [22] J. Luo, Y. Yang, Y. Yuan, S. Xu, and W. Hao, An imbalanced learning-based sampling method for physics-informed neural networks, *J. Comput. Phys.* **534**, 114010 (2025).
  - [23] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, *Acta Mech. Sin.* **37**, 1727 (2021).
  - [24] H. Hu, L. Qi, and X. Chao, Physics-informed neural networks (PINN) for computational solid mechanics: Numerical frameworks and applications, *Thin-Walled Struct.* **205**, 112495 (2024).
  - [25] M. Baldan, P. Di Barba, and D. A. Lowther, Physics-informed neural networks for inverse electromagnetic problems, *IEEE Trans. Magn.* **59**, 1 (2023).
  - [26] Á. M. Soto, A. Cervantes, and M. Soler, Physics-informed neural networks for high-resolution weather reconstruction from sparse weather stations, *Open Res. Eur.* **4**, 99 (2024).
  - [27] A. Steane, Quantum computing, *Rep. Prog. Phys.* **61**, 117 (1998).
  - [28] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM Rev.* **41**, 303 (1999).
  - [29] A. H. S. Shayegan, Quantum linear solvers for scientific computing: A comparison of VQLS, HHL and quantum annealing on time-fractional diffusion problems, *Sci. Rep.* **16**, 10278 (2026).
  - [30] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles, Variational quantum linear solver, *Quantum* **7**, 1188 (2023).
  - [31] R. D. Somma, D. Nagaj, and M. Kieferová, Quantum speedup by quantum annealing, *Phys. Rev. Lett.* **109**, 050501 (2012).
  - [32] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, *Phys. Rev. Lett.* **103**, 150502 (2009).
  - [33] Y. Li, B. Peng, Y. Wang, Z. Zhang, and Y. Wang, HHL algorithm without ancilla postselection for Toeplitz and Hankel systems, *Phys. Lett. A* **531**, 130154 (2025).
  - [34] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature (London)* **549**, 195 (2017).
  - [35] Y. Xiao, L. Yang, C. Shu, S. Chew, B. Khoo, Y. Cui, and Y. Liu, Physics-informed quantum neural network for solving forward and inverse problems of partial differential equations, *Phys. Fluids* **36**, 097145 (2024).
  - [36] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, *Quantum Sci. Technol.* **4**, 043001 (2019).
  - [37] A. Farea, S. Khan, and M. Serdar Celebi, QCPINN: Quantum-classical physics-informed neural networks for solving PDEs, *Mach. Learn.: Sci. Technol.* **6**, 045053 (2025).
  - [38] S. Berger, N. Hosters, and M. Möller, Trainable embedding quantum physics informed neural networks for solving nonlinear PDEs, *Sci. Rep.* **15**, 18823 (2025).
  - [39] A. Setty, R. Abdusalamov, and F. Motzoi, Self-adaptive physics-informed quantum machine learning for solving differential equations, *Mach. Learn.: Sci. Technol.* **6**, 015002 (2025).
  - [40] F. Y. Leong, W.-B. Ewe, S. B. Q. Tran, Z. Zhang, and J. Y. Khoo, Hybrid quantum physics-informed neural network: Towards efficient learning of high-speed flows, *Comput. Fluids* **301**, 106782 (2025).
  - [41] M. A. Nabian, R. J. Gladstone, and H. Meidani, Efficient training of physics-informed neural networks via importance sampling, *Comput.-Aided Civ. Infrastruct. Eng.* **36**, 962 (2021).
  - [42] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, *SIAM Rev.* **63**, 208 (2021).
  - [43] S. Tian, Y. Yu, and B. Li, A residual-based adaptive refinement physics-informed neural networks (RAR-PINNs) method for fifth-order KdV equation, *Chin. Phys. B* **35**, 050201 (2025).
  - [44] K. Luo, J. Zhao, Y. Wang, J. Li, J. Wen, J. Liang, H. Soekmadji, and S. Liao, Physics-informed neural networks for PDE prob-



- lems: A comprehensive review, *Artif. Intell. Rev.* **58**, 323 (2025).
- [45] W. Wu, S. Duan, Y. Sun, Y. Yu, D. Liu, and D. Peng, Deep fuzzy physics-informed neural networks for forward and inverse PDE problems, *Neural Netw.* **181**, 106750 (2025).
- [46] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, *et al.*, PennyLane: Automatic differentiation of hybrid quantum-classical computations, [arXiv:1811.04968](https://arxiv.org/abs/1811.04968).
- [47] A. Paszke *et al.*, PyTorch: An imperative style, high-performance deep learning library, [arXiv:1912.01703](https://arxiv.org/abs/1912.01703).
- [48] T. D. Barve, A. Y. Samant, and M. S. Kulaye, A comparative study of optimization algorithms in deep learning: SGD, Adam, and beyond, *Int. J. Comput. Technol. Electron. Commun.* **8**, 11018 (2025).