Complex-valued physics-informed machine learning for efficient solving of quintic nonlinear Schrödinger equations

Lei Zhang,[1] Mengge Du,[1] Xiaodong Bai,[2] Yuntian Chen,[3,4,*] and Dongxiao Zhang[3,4,5,†]

[1]*College of Engineering, Peking University, Beijing 100871, China*

[2]*College of Physics and Hebei Key Laboratory of Photophysics Research and Application, Hebei Normal University, Shijiazhuang 050024, HeBei, China*

[3]*Ningbo Institute of Digital Twin, Eastern Institute of Technology, Ningbo, Zhejiang 315200, China*

[4]*Zhejiang Key Laboratory of Industrial Intelligence and Digital Twin, Eastern Institute of Technology, Ningbo, Zhejiang 315200, China*

[5]*National Center for Applied Mathematics Shenzhen (NCAMS), Southern University of Science and Technology, Shenzhen 518000, Guangdong, China*

(Received 23 October 2024; accepted 14 January 2025; published 13 February 2025)

The Gross-Pitaevskii equation (GPE), a specialized form of the nonlinear Schrödinger equation (NLSE), plays a pivotal role in quantum mechanics, optics, and condensed matter physics, modeling phenomena such as superfluidity, quantum turbulence, and solitons, while serving as a cornerstone for advancing the study of nonlinear wave propagation and its technological applications. In this paper, we propose a solver, Complex-Valued Physics Informed Neural Network (CV-PINN) for NLSEs using physics-informed learning machines with complex representation. This method integrates complex values and algebraic properties directly into the neural network, with its structure mirroring the computation process of complex numbers, thereby significantly enhancing its ability to effectively solve complex problems. Additionally, we introduce a collocation-point sampling method called Predictive Dynamic Monitoring Sampling (PDM sampling), which adaptively adjusts the distribution of collocation points during training based on the model's historical performance. We conducted extensive empirical evaluations of the CV-PINN and the PDM sampling using a series of NLSE/GPEs (a total of 16 solved examples). Compared to the traditional real-valued PINN, the CV-PINN demonstrated higher accuracy, faster convergence, greater robustness, and better stability in these cases. Moreover, the PDM sampling proved to be effective in preventing model degradation and significantly enhancing the model's convergence speed and predictive accuracy when compared to traditional sampling methods. This advancement provides an approach and perspective for solving complex partial differential equations, thereby offering insights for the broader application of PINNs across various fields.

DOI: 10.1103/PhysRevResearch.7.013164

I. INTRODUCTION

The Gross-Pitaevskii equation (GPE) [1], as a significant form of nonlinear Schrödinger equation (NLSE), holds profound importance in quantum mechanics, optics, and condensed matter physics. The GPE governs the evolution of the wave function in Bose-Einstein condensate (BEC) systems and is extensively applied to the study of phenomena such as superfluidity, quantum turbulence, and solitons. Its significance lies not only in advancing our understanding of quantum fluid dynamics but also in providing a foundation for exploring nonlinear wave propagation. This is particularly relevant in cutting-edge technological domains, including optical fiber communication, ultracold atomic gases, and quantum computing. Thus, accurately solving the GPE is essential for in-depth investigations of these physical systems and their practical applications.

Traditionally, nonlinear Schrödinger equations are solved using advanced numerical methods such as finite-difference methods [2], finite-element methods, and spectral methods [3,4]. These approaches are capable of delivering high-precision numerical solutions and are particularly effective in handling systems with complex boundary conditions and nonlinear interactions. However, when applied to scenarios with high nonlinearity, achieving accurate solutions requires finer discretization in both time and space, leading to a significant increase in computational complexity [5]. For example, the split-step Fourier method, while offering high accuracy and efficiency under certain conditions, exhibits exponentially increasing computational demands as the time and space steps are refined. Consequently, traditional numerical methods become inefficient and often impractical for high-dimensional or strongly nonlinear problems. Therefore, it is imperative to

*Contact author: ychen@eitech.edu.cn
†Contact author: dzhang@eitech.edu.cn

develop more efficient and flexible computational strategies to address these challenges.

In recent years, advances in deep learning and machine learning have provided new approaches to addressing these challenges. Among them, Physics-Informed Neural Networks (PINNs) have garnered significant attention within the scientific community due to their ability to seamlessly incorporate physical laws into the neural network framework [6–8]. PINNs were first introduced by Raissi *et al.* in 2019 [9]. These networks are specifically designed to address the problems of solving differential equations by embedding governing physical equations into the network architecture as loss terms. They employ a unified computational framework that offers a novel and concise approach for solving high-dimensional and highly nonlinear partial differential equation (PDE) problems. This approach is applicable to a broad range of PDEs, encompassing both general PDEs [10], such as linear PDEs and constant-coefficient PDEs, and more complex forms, including complex-valued PDEs [11], variable coefficient PDEs [12,13], fractional-order PDEs [14], and partial differential-integral equations [15]. This efficiency endows PINNs with high applicability across a wide range of scientific domains where differential equations are commonly encountered [16,17].

This study aims to develop an efficient solver based on PINNs to tackle quintic nonlinear Schrödinger equations, as exemplified by the Gross-Pitaevskii equation. The solver is designed to simulate the spatiotemporal dynamics of bosons under various external potential fields, accounting for both two-body and three-body interactions. Our goal is to balance computational efficiency and solution accuracy, overcoming the computational complexity and time-intensive nature posed by traditional methods when solving such nonlinear complex partial differential equations. This also provides an efficient and precise framework for addressing these problems.

In existing studies utilizing PINNs to solve nonlinear Schrödinger equations, Raissi *et al.* initially applied the PINN framework to solve NLSEs with cubic nonlinear effects [9], while Jiang *et al.* extended this approach to explore solutions under varying initial conditions [18]. D. Liu *et al.* further expanded the application of PINN to two-dimensional NLSEs, achieving both forward and inverse solutions [19]. Additionally, Zhong *et al.* employed PINNs to investigate the Gross-Pitaevskii equation with complex $\mathcal{PT}$-symmetric potentials, simulating the dynamic evolution within a single atomic timescale [20]. These efforts have preliminarily demonstrated the effectiveness of PINNs in solving NLSEs [21] and GPEs. However, current studies still face two major limitations: they either focus on low-order systems (cubic NLSE) without the influence of external potential fields or are restricted to small spatiotemporal domains, thereby limiting the applicability of the models. Extensive studies have shown that as the nonlinearity of equations increases or the solution domains expand, traditional PINN models encounter significant optimization challenges [22–24]. These issues have driven researchers to explore new solution strategies. For instance, L. Li *et al.* proposed an adaptive gradient-enhanced physics-informed neural network to address the difficulty of remote evolution in large domains for generalized inhomogeneous NLSEs [25]. Meanwhile, Pu and Chen introduced a segmented PINN approach to tackle the challenges associated with long-time evolution in GPE solutions [11]. However, these studies are limited to systems with third-order nonlinearity. In contrast, our research focuses on solving the Gross-Pitaevskii equation with fifth-order nonlinear terms and additional potential-energy components. This type of complex system significantly increases the difficulty of training and optimizing PINN, making it challenging to achieve high accuracy and convergence. To the best of our knowledge, such problems have not been systematically addressed in existing studies, highlighting the complexity of our work.

To address these challenges, we first take into account the complex-valued nature of NLSEs and propose an innovative approach: the Complex-Valued Physics-Informed Neural Network (CV-PINN). In existing studies, researchers typically handle complex-valued partial differential equations (PDEs) by separating the real and imaginary parts and treating them as independent outputs of the neural network [11,20,26]. While this approach simplifies the problem, it may not fully capture the intrinsic characteristics of complex arithmetic, thereby limiting the accuracy and efficiency of the solution. Extensive research has demonstrated that the representations based on complex numbers in neural networks offer numerous advantages, including lower prediction errors [27], faster convergence and easier optimization [28], stronger robustness, and better generalization capabilities [29]. These performance benefits are particularly evident when dealing with complex-valued tasks as well as nonlinear inseparable real-valued tasks [30,31]. Although such complex-valued neural networks have not been applied in the PINN, Tiwari *et al.* introduced complex arithmetic into operator learning, parametrized the integral kernel in the complex fractional Fourier domain, and demonstrated its advantages over real-valued neural networks [32]. Inspired by this, we integrate complex representations into the PINN framework and embed complex arithmetic rules within the network structure to solve complex partial differential equations.

Second, to address the challenges PINNs face in solving highly nonlinear problems and large spatiotemporal domains, such as training difficulty and slow convergence, we propose a Predictive Dynamic Monitoring-based collocation-point sampling strategy (PDM sampling). Unlike existing methods that mitigate training challenges by merely partitioning the spatiotemporal domain or reallocating sampling points based on gradients or residuals [23,33–36], this study focuses on the dynamic evolution of the solution during the training process. By analyzing how the model's output evolves within the solution domain during the training process, we strategically increase the density of sampling points in regions with rapid variations, thereby achieving targeted regional optimization. This strategy significantly enhances the training efficiency and solution accuracy of the model.

Our contributions in this paper are as follows:

Deep learning-based GPEs solver: A rapid and precise approach for solving the evolution of particles in Bose-Einstein condensate (BEC) systems.

Complex-valued physics-informed learning machines: A PINN architecture that seamlessly integrates complex values and algebraic properties directly into its neurons, emulating the computation process of complex numbers. This

TABLE I. Parameter settings for the solved equations. Equation-1 considers two-body interactions. Equation-2 incorporates both two-body and three-body interactions. Equation-3 through Equation-7, in addition to two-body and three-body interactions, consider the effects of different external potentials. Specifically, Equation-3 includes the effects of a harmonic potential, Equation-4 through Equation-6 consider the combined effects of harmonic and periodic potentials, and Equation-7 takes into account a $\mathcal{PT}$-symmetric potential.

Equation	Description	g	κ	$V(x)$
1	Two-body interactions	2	0	0
2	Two-body and three-body interactions	2	2	0
3	Two/three-body interactions and Harmonic potentials	2	2	$0.2x^2$
4	Two/three-body interactions, Harmonic potentials and periodic potentials	2	2	$0.2x^2 + \cos(x)$
5	Two/three-body interactions, Harmonic potentials and periodic potentials	2	2	$0.2x^2 + \sin(x)$
6	Two/three-body interactions, Harmonic potentials and periodic potentials	2	2	$0.2x^2 + \cos^2(x)$
7	Two/three-body interactions and $\mathcal{PT}$-symmetric potential	2	2	$-\text{sech}^2(x) + i\text{sech}(x)\tanh(x)$

design offers enhanced generalization, robustness, faster convergence, and easier optimization for complex-valued problems.

Predictive dynamic monitoring sampling: A collocation point sampling strategy that dynamically monitors the training process, employing adaptive attention mechanisms to adjust point distribution in real time. This approach significantly enhances model convergence speed and predictive accuracy in PINNs.

The paper is organized as follows. In Sec. II, we define the problems and equations under discussion. Section III presents the methodology, where we propose the CV-PINN framework and introduce the predictive dynamic monitoring sampling strategy. Following that, Sec. IV details the specific training settings and evaluation metrics while also describing two real-valued PINNs used as baselines. Section V offers a detailed comparative study on the accuracy, convergence, and parameter efficiency of the proposed CV-PINN against the PINN through numerical examples, along with a discussion of the effectiveness of the PDM method with specific examples. Finally, Sec. VI concludes the paper.

II. PROBLEM DEFINITION

The equations we discuss are the dimensionless 1D Gross-Pitaevskii equations, which are usually used to describe typical 1D Bose-Einstein condensate (BEC).

In quantum system. BECs represent a unique state of matter formed at ultracold temperatures where particles coalesce into a single quantum state, exhibiting macroscopic quantum phenomena. The Gross-Pitaevskii equation is a fundamental mathematical model used to describe the behavior of BECs. Solving the GPE is crucial for understanding the dynamics, stability, and interactions within BECs, which have implications in quantum mechanics, superfluidity, and potential applications in quantum computing and precision measurement technologies.

In this paper, we employ PINNs to solve GPEs with specific forms, simulate and analyze the dynamic evolution of bosons, considering complex two-body and three-body interactions, as well as different external potentials such as harmonic traps, periodic potentials, and $\mathcal{PT}$-symmetric potentials [37]. The nonlinear interaction potential $U(x)$ in the GPE, accounting for both two-body and three-body interactions, can be expressed as

$$U(x) = g|\psi|^2 + \kappa|\psi|^4. \tag{1}$$

Then, the GPE can be written as

$$i\hbar\frac{\partial\psi}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2}\psi + g|\psi|^2 + \kappa|\psi|^4 + V(x)\psi, \tag{2}$$

where $\hbar$ is the reduced Planck constant; ψ is the wave function, a complex function describing the quantum state of particles, dependent on position x and time t; $g|\psi|^2 + \kappa|\psi|^4$ is the nonlinear interaction potential, g is the nonlinear coefficient related to two-body interactions, typically associated with the scattering length; κ is the nonlinear coefficient related to three-body interactions, which describes the strength of the three-body interactions; and $V(x)$ is the external potential, describing the electromagnetic trap needed for the experimental realization of a BEC. This equation is a higher-order nonlinear Schrödinger equation, often referred to as a quintic nonlinear Schrödinger equation.

The specific parameter settings [g, κ, and $V(x)$] for the solved equations are shown in Table I. We have computed their solutions over 5 atomic time units with Dirichlet boundary conditions $\psi(x = -20, t) = \psi(x = 20, t) = 0$ for three different initial conditions:

$$\psi_1(x, t = 0) = \exp(-x^2/10), \tag{3}$$

$$\psi_2(x, t = 0) = \frac{2\sin(x)}{\exp(-x) + \exp(-x)}, \tag{4}$$

$$\psi_3(x, t = 0) = \frac{2\cos(x)}{\exp(-x) + \exp(-x)}. \tag{5}$$

The reference solutions were obtained using the split-step Fourier spectral method. For the sake of discussion, we number the solved GPEs in the format GPE-{a-b}, where "a" represents the initial condition and "b" represents the equation.

III. METHODOLOGY

A. Physics-informed neural networks

PINNs typically embed physical equations by constructing a loss function, which generally consists of two parts: a term enforcing the physical laws and a traditional data-fidelity term (if applicable).

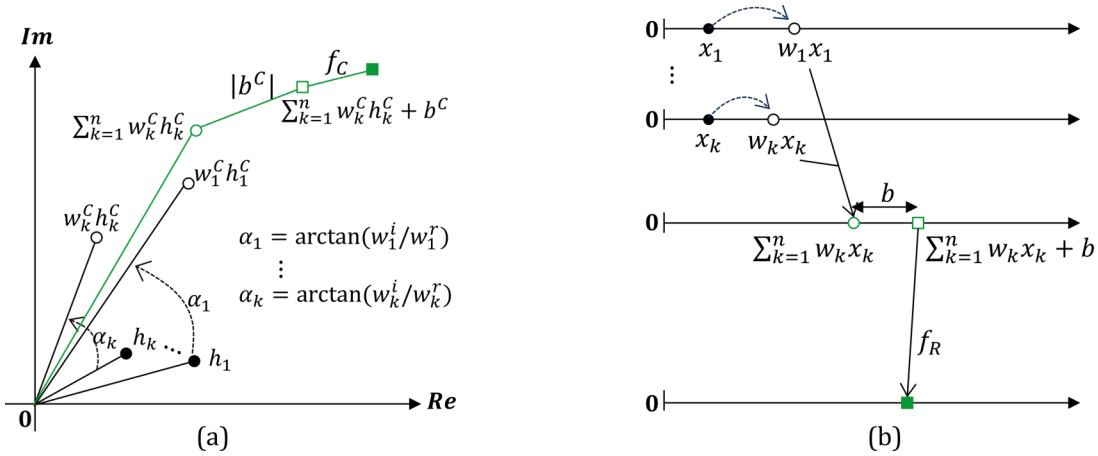

FIG. 1. Comparison between complex neuron and real neuron. Panel (a) illustrates the geometric interpretation of complex number operations. The complex number $h_k = u_k + iv_k$ is represented as a vector in the complex plane. When multiplied by the complex weight $w_k^c = w_k^r + iw_k^i$, the vector is rotated by an angle $\alpha = \arctan(w_k^r/w_k^i)$ and scaled by $|w_k^c|$, before the addition of another complex number b. Panel (b) contrasts real operations, where multiplication and addition occur in one-dimensional space. In contrast, complex number operations involve both rotation and scaling within the complex plane, transitioning from one-dimensional to two-dimensional space.

Consider a general PDE:

$$\begin{aligned} \partial_t u(t,x) + \mathcal{F}[u(t,x)] &= 0,\ (t,x) \in [0,T] \times \Omega \\ u(0,x) &= u_0(x),\ x \in \Omega \\ \mathcal{G}(u)(x,t) &= 0,\ x \in [0,T] \times \partial\Omega, \end{aligned} \tag{6}$$

where $\partial\Omega$ is the boundary of the domain Ω, $\mathcal{F}$ is a differential operator, and $\mathcal{G}$ denotes a boundary operator.

By minimizing the L^2 error, an approximate solution $f(t,x)$ can be found:

$$\begin{aligned} \mathcal{J}(f) = &\|\partial_t f + \mathcal{F} f\|_{2,[0,T]\times\Omega}^2 + \|\mathcal{G}\|_{2,[0,T]\times\partial\Omega}^2 \\ &+ \|f(0,\cdot) - u_0\|_{2,\Omega}^2 . \end{aligned} \tag{7}$$

In PINNs, approximate solution f is obtained via a neural network $\Psi_{\mathrm{NN}}^{\theta}(x,t)$, and $\mathcal{J}(f)$ is transformed into the loss term of the neural network. Once sufficiently optimized, $\Psi_{\mathrm{NN}}^{\theta}$ serves as a surrogate PDE solver, providing reliable approximate solutions.

B. Complex-valued neural networks

Complex-Valued Neural Networks (CVNNs) offer a natural extension to traditional real-valued neural networks by incorporating complex numbers, enabling direct processing of both phase and amplitude information. This capability allows CVNNs to capture the essential attributes of two-dimensional data, such as rotations, similarity transformation, and parallel displacement of straight lines, circles, etc., which are often inadequately represented in traditional real neural networks (Fig. 1) [38]. This characteristic renders CVNNs particularly useful in fields such as telecommunications [39,40], speech processing [41,42], and remote sensing [43].

In CVNNs, a complex neuron is represented by logically distinct real-valued entities, where the complex weight $w = w_r + iw_i$ and complex bias $b = b_r + ib_i$ are simulated using real-valued arithmetic to perform complex operations. Consider a complex-valued fully connected layer that has m neurons, for an n-dimensional complex input $\mathbf{h} = \mathbf{u} + i\mathbf{v}$; if we use matrix notation to represent real and imaginary parts of the complex operation, its weights and bias can be represented as

$$\mathbf{w} = \mathbf{w}_r + i\mathbf{w}_i, \tag{8}$$

$$\mathbf{b} = \mathbf{b}_r + i\mathbf{b}_i. \tag{9}$$

The output is

$$[\Re(\mathbf{w}\cdot\mathbf{h})\Im(\mathbf{w}\cdot\mathbf{h})] = [\mathbf{u}\ \ \mathbf{v}]\cdot\begin{bmatrix} \mathbf{w}_r & \mathbf{w}_i \\ -\mathbf{w}_i & \mathbf{w}_r \end{bmatrix} + [\mathbf{b}_r \quad \mathbf{b}_i], \tag{10}$$

where $i = \sqrt{-1}$ is known as the complex unit; $\mathbf{u},\ \mathbf{v} \in \mathbb{R}^n$ are the real and imaginary parts of input $\mathbf{h} \in \mathbb{C}^n$; $\mathbf{w}_r,\ \mathbf{w}_i \in \mathbb{R}^{nm}$ represent the real and imaginary components of the weight $\mathbf{w} \in \mathbb{C}^{\mathbf{nm}}$; and $\mathbf{b}_r,\ \mathbf{b}_i \in \mathbb{R}^m$ are the corresponding parts of the bias $\mathbf{b} \in \mathbb{C}^m$. And, we can find that *Real-part Neuron* and *Imaginary-part Neuron* influence each other via their weights [44].

When incorporating activation functions, they should either be a fully complex activation functions or a split-type activation function composed of two real-valued functions.

C. Complex-valued PINN

We propose a complex-valued PINN based on physics-informed neural networks and complex neural networks.

1. Model architecture

In the context of solving the GPEs, the model inputs are real valued: the spatial coordinate x and the temporal coordinate t. Therefore, in addition to replacing all neurons in the traditional PINN with complex neurons, it is also necessary to consider the connection between real-valued inputs and complex-valued outputs. Generally, when a complex layer receives real-valued inputs, the real inputs must be augmented to form a complex number, either by setting their imaginary parts to zero or by duplicating the real parts into the imaginary parts. Research indicates that the absence of imaginary

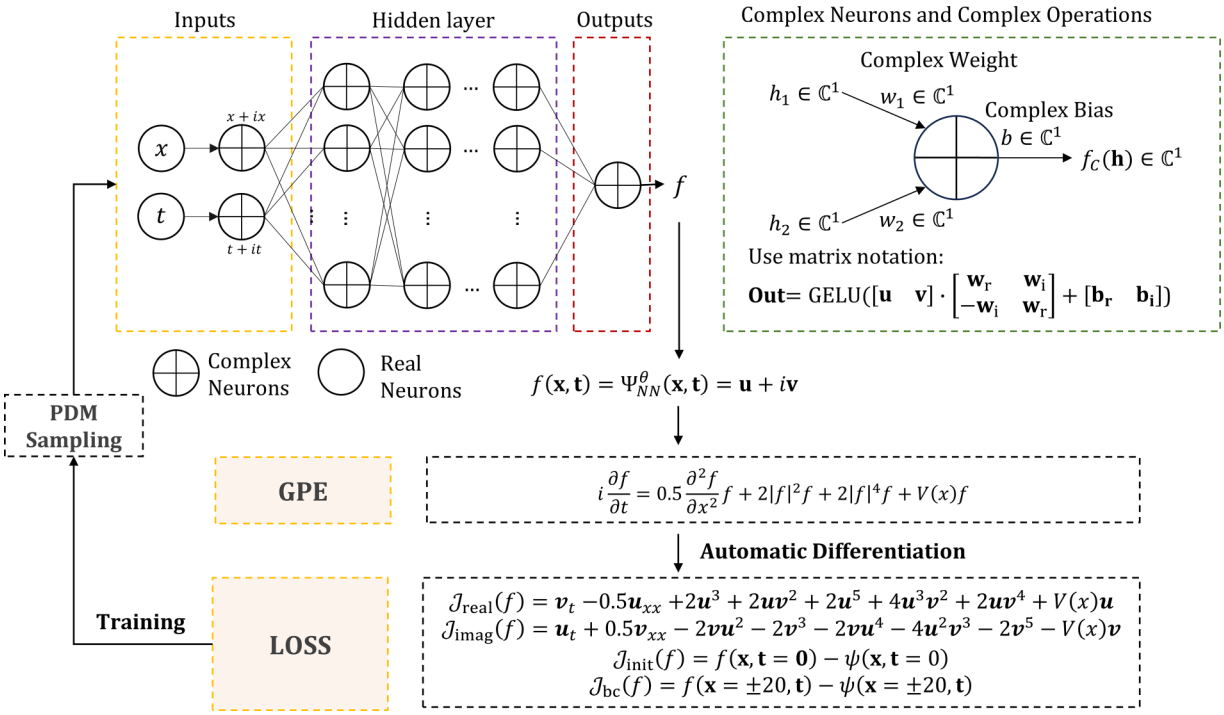

FIG. 2. Schematic of the CV-PINN framework. Structurally, CV-PINN is a fully connected neural network composed of complex neurons. It accepts real inputs, extends them in the imaginary part to form complex inputs, and then passes these inputs to the complex neurons. The tensors between each neuron follow the rules of complex arithmetic. The model outputs a complex number, which is an approximate solution to the wave function. Relying on the automatic differentiation of the neural network, the corresponding PDE residuals can be easily obtained and included as part of the model loss during training.

components in the inputs can cause a lag in the training of the imaginary parts within the neural network, thereby reducing the predictive accuracy of complex-valued neural networks [45]. Considering this, we duplicate the real part of the inputs into the imaginary part when handling real-valued inputs. This transforms the real inputs into complex inputs, allowing for complex arithmetic operations within the complex layers (Fig. 2).

In our framework, the CV-PINN structurally consists of 1 input layer, 4 hidden layers each containing 50 complex neurons, and 1 output layer with 1 complex neuron, forming a fully connected structure (the model architecture is determined based on ablation experiments). It is important to emphasize that we represent the real part w_r and the imaginary part w_i of a complex number as logically distinct real-valued entities and simulate complex arithmetic using real-valued arithmetic internally. Specifically, when using a fully connected layer with 50 complex neurons, which receives an input in the form of a 1×50 complex matrix, the weight tensor of this layer should be of size 100×100, adhering to the following structure:

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_r & \mathbf{w}_i \\ -\mathbf{w}_i & \mathbf{w}_r \end{bmatrix}. \tag{11}$$

2. Activation functions

When applying activation functions in complex neural networks, one important consideration is that the loss function and activations should be differentiable with respect to the real and imaginary parts of each complex parameter in the network. This means that activation functions should be constrained to be complex differentiable or holomorphic, although using holomorphic functions allows one to share gradient values (because the activation satisfies the Cauchy-Riemann equations). However, finding an activation function that is holomorphic over the entire domain is not straightforward. Research [46] indicates that enforcing holomorphic is an unnecessary restriction. Applying differentiable activation functions separately to the real and imaginary parts can ensure the effectiveness of backpropagation on both parts. In this work, we apply a split-type activation function: C-GELU. It applies GULE activation function to both the real and imaginary parts of the hidden layers. This can be expressed as

$$outputs = [\mathrm{GELU}(\Re(\mathbf{W} \cdot \mathbf{h})) \quad \mathrm{GELU}(\Im(\mathbf{W} \cdot \mathbf{h}))]. \tag{12}$$

This activation function satisfies the Cauchy-Riemann equations when the real and imaginary parts are strictly positive or negative.

3. Loss terms

The core idea of PINNs is to directly embed PDEs into the neural network. When extending PINNs to the complex domain to solve complex-valued PDEs, it is crucial to properly compute the derivatives of complex solutions. It is important to note that although the neural network deals with complex values, the underlying operations are still based on real numbers. Current deep learning frameworks, such as PyTorch, do not support the direct computation of derivatives for complex outputs. Therefore, to leverage automatic differentiation

techniques to obtain the derivatives of complex outputs, it is necessary to compute the residuals of the PDE separately for the real and imaginary parts. This approach ensures that PINNs can be applied effectively and accurately within the complex domain.

For example, consider the GPE with two-body and three-body interactions under a harmonic potential. The original PDE residuals are given by

$$\mathcal{J}(f) = i\frac{\partial f}{\partial t} - 0.5\frac{\partial^2 f}{\partial x^2} f + 2|f|^2 f + 2|f|^4 f + 0.2x^2 f, \tag{13}$$

$$f = u + iv, \tag{14}$$

where f is the output of the PINN network; u and v are the real and imaginary parts of f. Thus, the real-part residual and the imaginary-part residual can be expressed as

$$\begin{aligned}\mathcal{J}_{\text{real}}(f) &= v_t - 0.5u_{xx} + 2u^3 + 2uv^2 + 2u^5 \\ &\quad + 4u^3v^2 + 2uv^4 + 0.2x^2u, \end{aligned}\tag{15}$$

$$\begin{aligned}\mathcal{J}_{\text{imag}}(f) &= u_t + 0.5v_{xx} - 2vu^2 - 2v^3 - 2vu^4 \\ &\quad - 4u^2v^3 - 2v^5 - 0.2x^2v. \end{aligned}\tag{16}$$

During the training process of the CV-PINN, the objective is to find solutions that minimize both loss terms (i.e., to reduce the modulus squared of the complex PDE residual to zero). This approach essentially transforms the problem of solving a single equation into the task of addressing a system of coupled equations. Therefore, in our framework, the loss function is composed of initial conditions, boundary conditions, and equation residuals, each containing both real and imaginary parts (fidelity data are not used).

$$\begin{aligned}\mathcal{L}_{\text{CV-PINN}}\left(\Psi^\theta_{\text{CVNN}}\right) &= \lambda_{\text{IC}}\frac{1}{N_{\text{IC}}}\sum_{i=0}^{N_{\text{IC}}}\left(\left|\text{Real}\left(\Psi^\theta_{\text{CVNN}}\left(x^i_{\text{IC}}, t^i_{\text{IC}}\right)\right) - u^i_{\text{IC,real}}\right|^2 + \left|\text{Imag}\left(\Psi^\theta_{\text{CVNN}}\left(x^i_{\text{IC}}, t^i_{\text{IC}}\right)\right) - u^i_{\text{IC,imag}}\right|^2\right) \\ &+ \lambda_{\text{BC}}\frac{1}{N_{\text{BC}}}\sum_{i=0}^{N_{\text{BC}}}\left(\left|\mathcal{G}\left(\Psi^\theta_{\text{NN}}\right)_{\text{Real}}\left(x^i_{\text{BC}}, t^i_{\text{BC}}\right)\right|^2 + \left|\mathcal{G}\left(\Psi^\theta_{\text{NN}}\right)_{\text{Imag}}\left(x^i_{\text{BC}}, t^i_{\text{BC}}\right)\right|^2\right) \\ &+ \lambda_f\frac{1}{N_f}\sum_{i=0}^{N_f}\left(\left|\text{Real}\left(\partial_t\Psi^\theta_{\text{NN}}\left(x^i_f, t^i_f\right)\right) + \mathcal{F}_{\text{Real}}\left[\Psi^\theta_{\text{NN}}\left(x^i_f, t^i_f\right)\right]\right|^2\right. \\ &+ \left.\left|\text{Imag}\left(\partial_t\Psi^\theta_{\text{NN}}\left(x^i_f, t^i_f\right)\right) + \mathcal{F}_{\text{Imag}}\left[\Psi^\theta_{\text{NN}}\left(x^i_f, t^i_f\right)\right]\right|^2\right). \end{aligned}\tag{17}$$

In this setup, the weighting coefficients for the initial conditions (IC) and equation residuals λ_f are set to 100, while the weighting coefficient for the boundary conditions λ_{BC} is set to 1. This configuration is intended to force the neural network to place greater emphasis on the initial conditions and equation residuals. Such values were empirically determined based on multiple experiments.

D. Predictive dynamic monitoring sampling

The selection of collocation points is crucial for the successful training of PINNs. Generally, we use sampling methods like Latin hypercube sampling (LHS) to perform uniform random sampling within the defined domain, which usually yields satisfactory solutions. However, when the solutions of PDEs exhibit rapid variations or steep gradients, or when the PDEs themselves are highly nonlinear, uniform sampling methods may fail to effectively capture the true solutions in these regions. This not only reduces training efficiency but can also lead to performance degradation during training. In such cases, exploring and implementing more refined sampling strategies becomes particularly important. In this regard, considerable research efforts have already been undertaken. For example, empirically distributing the number of collocation points within sampling regions [47], adaptive weighting of collocation points during training based on the loss or PDE residuals at each point [35,36], and adding more collocation points in specific regions based on the solution's gradient or residuals [23,33,34]. These methods aim to increase the quantity or weight of collocation points in regions that are challenging to train or exhibit poor performance, thereby enhancing the model's focus on these areas during training and guiding it towards more accurate solutions.

However, the aforementioned strategies typically consider the solution or residual distribution at a specific moment after a certain amount of training, without accounting for the dynamic nature of the training process. Given that network training is a dynamic process, we believe that the regions requiring focused training should be determined dynamically. Based on this consideration, we propose a collocation-point sampling method based on PDM sampling. This method not only focuses on the model's current performance but also leverages the historical performance during training to dynamically analyze and adjust the distribution of sampling points. Specifically, the method dynamically adjusts the sampling density and locations by evaluating the magnitude of changes in loss or solution during training. This ensures that the model can effectively and promptly adapt to the complex variations of the solution, as illustrated in Fig. 3.

Additionally, in extensive experiments and preliminary works [48], we have observed that the training process of PINNs typically encompasses three stages when solving strongly nonlinear equations or when the temporal domain is long and the spatial domain is large:

(i) *Preheating period:* During this phase, the loss associated with the initial conditions hardly decreases, and the

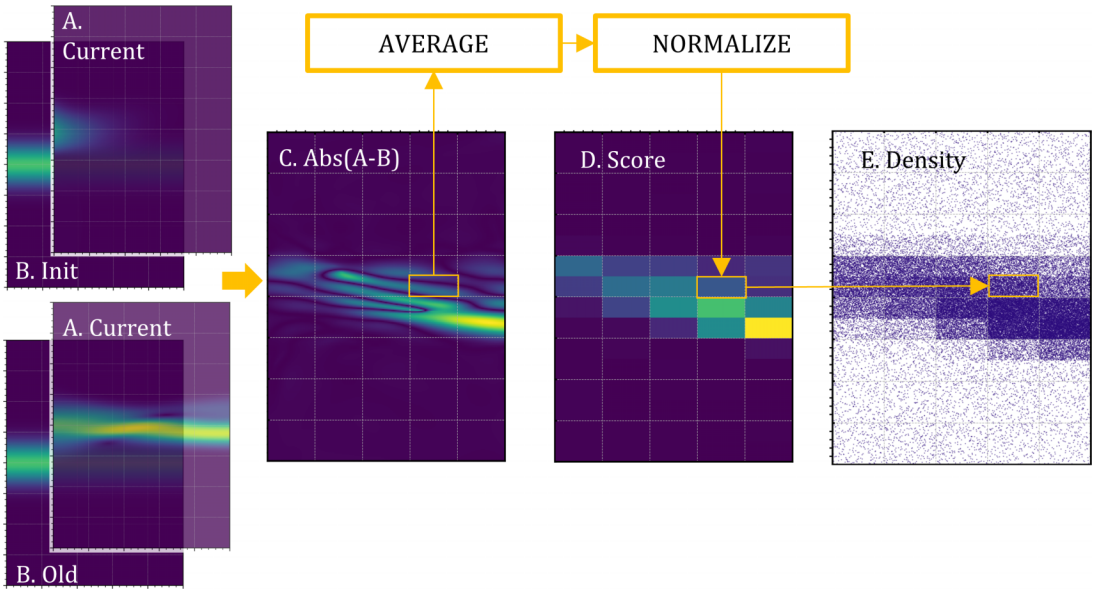

FIG. 3. PDM-sampling process. The process begins with a forward computation to obtain solutions at the specified nodes before and after (A and B), then takes the absolute difference between the two (C) and computes the average differences within each grid (D). The resulting difference matrix is smoothed and normalized to create a score matrix, where each grid's score corresponds to the percentage of collocation points it occupies. Finally, points are sampled within each grid according to their scores using the LHS method, with the resulting set of points (E) serving as collocation points for model training.

model tends to converge to trivial solutions. The duration of this stage varies significantly depending on the equation and initial-boundary conditions. For example, in our cases, the GPE-21 exits this stage after about 100 epochs, whereas the GPE-13 remains in this stage for 30 000 to 40 000 epochs, as illustrated in Fig. 4. In some scenarios, the model may get stuck in this stage, leading to training failure and model deterioration.

(ii) *Rapid descent period:* At a certain point in the training, the loss associated with the initial conditions begins to decrease rapidly. This phase is often accompanied by an initial increase followed by a decrease in PDE residuals, indicating

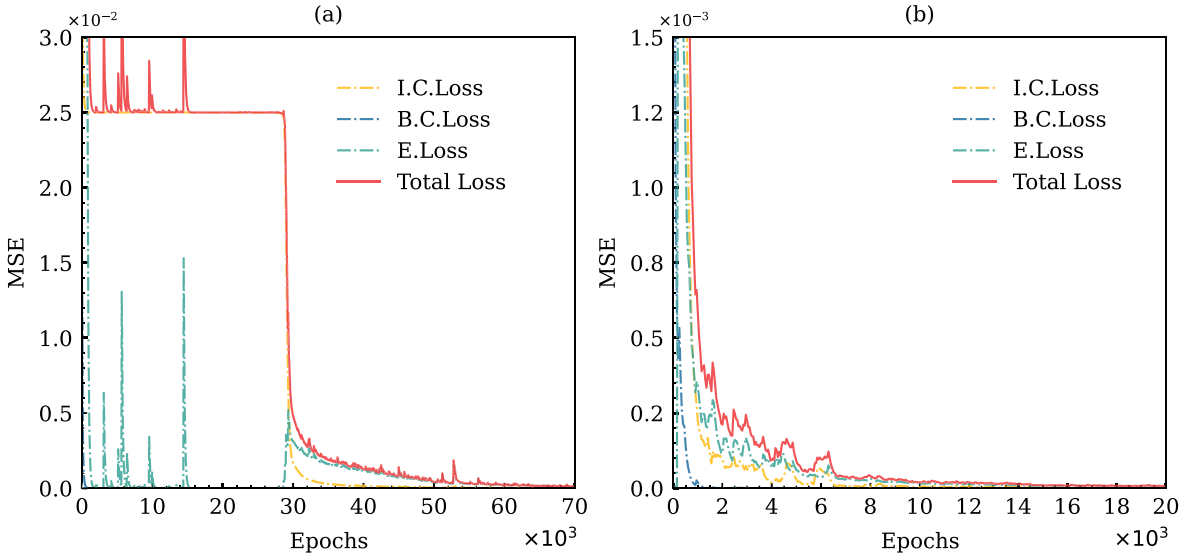

FIG. 4. Loss-convergence curves for GPE-13 and GPE-21. This figure illustrates the loss-convergence curves for solving GPE-13 (a) and GPE-21 (b). The terms I.C. Loss, B.C. Loss, and E Loss correspond to the initial condition loss, boundary condition loss, and equation residual loss, respectively. In panel (a), the training for GPE-13 exhibits a prolonged initial phase, with the initial condition loss starting to decrease only after nearly 30 000 iterations. During this phase, the high initial condition loss alongside near-zero values for other loss components indicates that the solution is a trivial one, essentially zero everywhere. In contrast, panel (b) shows that for GPE-21, the initial condition loss becomes very low in fewer than 1000 iterations. This discrepancy highlights the temporary "degradation" phenomenon discussed in the text, which occurs in models for more complex equations.

ALGORITHM 1. Training with PDM sampling.

Inputs: PINN with parameters θ as $\Psi_{\mathrm{NN}}^{\theta}$; grid dimensions $W \times L$; evaluation points numbers $m \times n$; A list of resampling iteration points P specifying when resampling occurs (e.g., [2000,3000,4000]); A normalization-satisfying smoothing kernel $K \in \mathbb{R}^{(k,k)}$; The total number of sampling points N.

Begin

1. Uniformly sampling $m \times n$ points $\mathrm{x_e} \in \mathbb{R}^{(m,n,2)}$ in the solution domain
2. Randomly sampling N collocation points $\mathrm{x}_f \in \mathbb{R}^{(N,2)}$ by LHS and train the PINN for a certain number of iterations
3. Compute $\Psi_{\mathrm{NN}}^{\theta}(\mathrm{x_e}) \in \mathbb{C}^{(m,n)}$ as ψ_{old}
4. Copy the initial condition $\psi(x, t = 0)$ along the t axis as $\psi_{\mathrm{new}} \in \mathbb{C}^{(m,n)}$
5. Differencing: $d = ||\psi_{\mathrm{new}}| - |\psi_{\mathrm{old}}||$, $d \in \mathbb{R}^{(m,n)}$
6. Averaging: $D_{i,j} = \frac{W \cdot L}{m \cdot n} \sum_{a=\frac{m}{W}(i-1)+1}^{\frac{m}{W} i} \sum_{b=\frac{n}{L}(j-1)+1}^{\frac{n}{L} j} d_{a,b}$, $D \in \mathbb{R}^{(W,L)}$
7. Smoothing: $D'_{i,j} = \sum_{a=1}^{k} \sum_{b=1}^{k} K_{a,b} D_{(i+a-1, j+b-1)}$, $D' \in \mathbb{R}^{(W,L)}$
8. Normalizing: $S = \frac{D'}{\sum_{a=1}^{W} \sum_{b=1}^{L} D'_{a,b}}$, $S \in \mathbb{R}^{(W,L)}$
9. Sampling: use LHS to sample $N \cdot S_{i,j}$ points per grid as the $\mathrm{x}_f \in \mathbb{R}^{(N,2)}$
10. $\psi_{\mathrm{old}} = \psi_{\mathrm{new}}$.
11. Training: train the PINN until the iteration reaches the next resampling node.
12. Repeat
13. Compute $\Psi_{\mathrm{NN}}^{\theta}(\mathrm{x_e}) \in \mathbb{C}^{(m,n)}$ as ψ_{new}
14. Run 5–11
15. until model converges or reaches the maximum number of iterations.

that the model is starting to move away from trivial solutions and the total loss drops swiftly.

(iii) *Convergence period:* All loss components decrease slowly and synchronously until convergence.

Based on the aforementioned three training stages, when employing the PDM sampling for resampling and training, we divide the process into two main phases: the *preheating phase* and the *evolution phase*. The primary goal of the preheating phase is to force the initial condition loss term to decrease, thereby driving the model away from trivial solutions and preventing degradation. This is achieved by using the initial conditions as the "sampling reference." In the second phase, the initial conditions are discarded, and resampling is carried out entirely based on the model's predictive results. The specific steps for PDM sampling are outlined in Algorithm 1.

This sampling method focuses collocation points on regions where the model's performance exhibits significant changes by monitoring the dynamic variations of the solution during training. This allows the network to achieve faster and better convergence. It is important to note that in the subsequent experiments, this resampling method is only applied to cases where conventional sampling methods struggle with training. The effectiveness of this approach is analyzed in Sec. V D.

IV. EXPERIMENTAL DETAILS

A. Training setup

In our study, we simulated the evolution of particles over 5 atomic time units for each GPE. The number of collocation points was set to 80 000. When resampling the collocation points using the PDM sampling, the solution domain was divided into 5 equal parts along the time and 16 equal parts along the spatial, resulting in a total of 80 grids. We resample the collocation points every 2500 to 10 000 training iterations, with the exact setting varying depending on the specific equation. Model training employed the Adam optimizer, along with the ReduceLROnPlateau learning-rate scheduler for dynamically adjusting the learning rate (the initial learning rate is set to 0.01, evaluated every 200 epochs. If the metric does not improve, the learning rate is reduced by a factor of 0.98). Additionally, to prevent gradient explosion during training, gradient clipping was applied before each update.

For each case study, the required training time for the model varies depending on the complexity of the governing equation. Generally, a satisfactory solution can be achieved after approximately 50 000 epochs, which typically takes about 2 h to complete (on a single Nvidia 3090 GPU). Once the model is fully trained, the inference time for predicting solutions at any given time or spatial location is reduced to the millisecond level.

B. Performance metrics

In this paper, the performance was quantified using three commonly used metrics: R^2, mean absolute error (MAE), and mean-squared error (MSE). Additionally, we use the discrepancy ratio (DR) to quantify the difference between the two models, which is calculated using the following formula:

$$\mathrm{DR_{MAE}} = \frac{\mathrm{MAE_B} - \mathrm{MAE_E}}{\mathrm{MAE_B}}, \tag{18}$$

$$\mathrm{DR_{MSE}} = \frac{\mathrm{MSE_B} - \mathrm{MSE_E}}{\mathrm{MSE_B}}, \tag{19}$$

where $\mathrm{MAE_B}$ and $\mathrm{MAE_E}$ represent the mean absolute errors of the baseline and experimental models, respectively. Similarly, $\mathrm{MSE_B}$ and $\mathrm{MSE_E}$ denote the mean-squared errors of the baseline and experimental models. A positive value indicates that the experimental model outperforms the baseline model, with higher values signifying a greater disparity between the two models.

TABLE II. Comparison of prediction performance between single and double real-valued PINN models. The table shows the prediction results for 2 models across 2 equations (GPE-12, GPE-13). Better results are highlighted in bold. It is important to note that an R^2 value of $1.000_{\pm 3.00\text{E}-06}$ does not imply $R^2 > 1$, but rather that it is extremely close to 1; hence, it is rounded to 1.000.

Equation	Model	MAE ($\times 10^{-3}$)	DR_{MAE}	MSE ($\times 10^{-5}$)	DR_{MSE}	R^2
GPE-12	**S**	$\mathbf{0.49_{\pm 0.01}}$	16.6%	$\mathbf{0.06_{\pm 0.01}}$	19.6%	$\mathbf{1.000_{\pm 3.00E-06}}$
	D	$0.60_{\pm 0.05}$		$0.09_{\pm 0.01}$		$1.000_{\pm 6.88\text{E}-06}$
GPE-13	**S**	$\mathbf{3.92_{\pm 0.88}}$	59.2%	$\mathbf{11.80_{\pm 4.50}}$	82.1%	$\mathbf{0.994_{\pm 2.18E-03}}$
	D	$9.36_{\pm 0.87}$		$62.00_{\pm 12.80}$		$0.966_{\pm 7.81\text{E}-03}$

S: Single PINN; D: Double PINN; Each set of experiments was repeated three times.

C. Baseline model

To validate the performance of the CV-PINN, we constructed two common real-valued PINN models for solving complex problems as comparisons:

1. Single real-valued PINN (model S)

This model has 2 inputs, 4 hidden layers, and 2 outputs, where the 2 outputs represent the real and imaginary parts of the target wave function, respectively. To ensure a valid comparison, model A adopts the same 4-layer architecture as CV-PINN. The number of nodes in each hidden layer of the PINN was adjusted to match the number of trainable parameters in each layer as closely as possible to the corresponding layer in CV-PINN. Notably, this adjustment in the number of parameters is based on the following relationship:

$$n^i_{\text{RV}} \times n^{i+1}_{\text{RV}} + n^{i+1}_{\text{RV}} = 2\left[\left(n^i_{\text{CV}} \times n^{i+1}_{\text{CV}}\right) + n^{i+1}_{\text{CV}}\right], \tag{20}$$

where n^i_{RV} represents the number of neurons in the ith layer of the PINN, and n^i_{CV} represents the number of neurons in the ith layer of the CV-PINN. The left side of the equation represents the number of parameters in PINN, while the right side represents the number of parameters in CV-PINN, using a single-layer network as an example.

When the CV-PINN network architecture is [2, 50, 50, 50, 50, 1], the corresponding PINN network architecture is [2, 70, 72, 71, 70, 2]. The real-valued models use the GELU activation function, too.

2. Double real-valued PINN (model D)

This model employs two separate neural networks, each receiving the same input information x and t, and each producing one output. One network specifically predicts the real part of the wave function, while the other network predicts the imaginary part. The architecture for each network is as follows: [2, 50, 50, 50, 50, 1]. Although these two networks independently handle the real and imaginary parts of the prediction, they are jointly trained using a common loss function. This design enables the networks to share learning signals and gradients during backpropagation, ensuring coordinated training and enhancing the overall coherence and consistency of the predictions.

Aside from the differences in network architecture, the PINN and the CV-PINN are kept identical in all other settings. This includes, but is not limited to, learning rate, optimizer configuration, training iterations, and any regularization strategies.

To compare the performance of the two models, we used GPE-12 and GPE-13 as examples; see Table II. The results indicate that single model outperforms double model. Therefore, in the subsequent experiments, we will only compare single model with the proposed CV-PINN.

V. NUMERICAL EXPERIMENTS AND RESULTS

A. Accuracy

We evaluated the solving capabilities of CV-PINN and PINN for complex-valued PDEs across 16 cases, encompassing 7 different GPEs and 3 initial conditions. The performance was quantified using three metrics: MAE, MSE, and R^2, along with two discrepancy ratios (DR_{MAE} and DR_{MSE}), with PINN serving as the baseline.

As shown in Table III, in the 16 cases tested, CV-PINN achieved the best performance in 15 out of 16. This indicates that the CV-PINN model demonstrates higher overall accuracy and stability compared to the PINN model, as evidenced by lower average MAE (3.27×10^{-3} vs 6.22×10^{-3}) and MSE values (6.03×10^{-5} vs 68.30×10^{-5}) across the repeated experiments, as well as lower standard deviation. The trends observed in the data are more clearly visualized in Fig. 5, where CV-PINN consistently maintains lower errors (indicated by smaller areas enclosed by the radar plot) in the comparison of 16 cases. Conversely, the PINN not only exhibits larger areas on the radar plots but also shows significantly larger errors in specific cases. This phenomenon is particularly pronounced in the comparison of MSE metrics, suggesting that the PINN model tends to produce predictions with a greater number of large error points. Although the findings are largely favorable, an outlier was identified in case GPE-12, where the DR_{MAE} values are slightly less than zero for the C (Complex PINN) model compared to the R (Real PINN) model. Despite this, the average MAE and MSE values are nearly identical after rounding (0.371×10^{-3} vs 0.366×10^{-3} for MAE, and 0.04×10^{-5} vs 0.04×10^{-5} for MSE) owing to the more precise calculations prior to rounding. This indicates that both PINN and CV-PINN provide high-precision solutions. Similar patterns were also observed in cases GPE-11, GPE-21, and GPE-32.

Additionally, another noticeable trend is that as the complexity of the equations increases [with more complex $V(x)$ and more rapidly varying initial conditions], the CV-PINN demonstrates significant advantages. In some cases, the error of the CV-PINN is even an order of magnitude lower than that of the PINN, and the lower standard error also indicates that

TABLE III. Performance comparison between CV-PINN and PINN. The table presents the predictive performance metrics for CV-PINN and PINN models across different equations. Better results are highlighted in bold. All results are derived from the mean values of three independent repeated experiments. It is also important to note that an R^2 value of $1.000_{\pm 1.19E-07}$ does not imply $R^2 > 1$, but rather that it is extremely close to 1; hence, it is rounded to 1.000.

Equation	Model	MAE ($\times 10^{-3}$)	DR_{MAE}	MSE ($\times 10^{-5}$)	DR_{MSE}	R^2
GPE-11	**C (ours)**	$\mathbf{0.31_{\pm 0.01}}$	6.8%	$\mathbf{0.03_{\pm 0.00}}$	7.4%	$1.000_{\pm 1.19E-07}$
	R	$0.33_{\pm 0.03}$		$0.03_{\pm 0.00}$		$1.000_{\pm 2.03E-06}$
GPE-21	**C (ours)**	$\mathbf{2.05_{\pm 0.08}}$	3.4%	$\mathbf{0.80_{\pm 0.09}}$	8.2%	$\mathbf{1.000_{\pm 5.37E-05}}$
	R	$5.90_{\pm 0.73}$		$9.18_{\pm 2.31}$		$0.994_{\pm 1.53E-03}$
GPE-12	**C (ours)**	$0.37_{\pm 0.01}$	−1.4%	$0.04_{\pm 0.00}$	0.2%	$1.000_{\pm 1.84E-07}$
	R	$0.37_{\pm 0.01}$		$0.04_{\pm 0.00}$		$1.000_{\pm 3.17E-07}$
GPE-32	**C (ours)**	$\mathbf{5.73_{\pm 0.86}}$	1.7%	$\mathbf{8.14_{\pm 2.50}}$	4.6%	$\mathbf{0.994_{\pm 1.67E-03}}$
	R	$5.90_{\pm 0.73}$		$9.18_{\pm 2.31}$		$0.994_{\pm 1.53E-03}$
GPE-13	**C (ours)**	$\mathbf{1.63_{\pm 0.31}}$	48.4%	$\mathbf{2.02_{\pm 0.78}}$	67.5%	$\mathbf{0.999_{\pm 3.63E-04}}$
	R	$5.09_{\pm 2.09}$		$31.60_{\pm 22.6}$		$0.982_{\pm 1.35E-02}$
GPE-23	**C (ours)**	$\mathbf{4.61_{\pm 1.02}}$	76.6%	$\mathbf{10.80_{\pm 5.40}}$	94.3%	$\mathbf{0.995_{\pm 2.86E-03}}$
	R	$31.50_{\pm 16.10}$		$826.00_{\pm 681.00}$		$0.721_{\pm 1.93E-01}$
GPE-33	**C (ours)**	$\mathbf{5.11_{\pm 1.35}}$	12.4%	$\mathbf{12.10_{\pm 6.77}}$	19.7%	$\mathbf{0.994_{\pm 3.33E-03}}$
	R	$5.74_{\pm 0.63}$		$13.70_{\pm 3.09}$		$0.993_{\pm 1.45E-03}$
GPE-14*	**C (ours)**	$\mathbf{1.77_{\pm 0.23}}$	49.3%	$\mathbf{2.05_{\pm 0.67}}$	72.3%	$\mathbf{0.999_{\pm 3.21E-04}}$
	R	$3.67_{\pm 0.80}$		$8.20_{\pm 3.17}$		$0.996_{\pm 1.54E-03}$
GPE-24*	**C (ours)**	$\mathbf{3.08_{\pm 0.12}}$	11.0%	$\mathbf{3.97_{\pm 0.39}}$	21.4%	$\mathbf{0.998_{\pm 1.96E-04}}$
	R	$3.56_{\pm 0.52}$		$6.19_{\pm 2.43}$		$0.997_{\pm 1.25E-03}$
GPE-34*	**C (ours)**	$\mathbf{6.05_{\pm 0.57}}$	44.3%	$\mathbf{14.10_{\pm 2.87}}$	68.4%	$0.993_{\pm 1.50E-03}$
	R	$11.90_{\pm 2.43}$		$70.10_{\pm 26.40}$		$0.960_{\pm 1.56E-02}$
GPE-15*	**C (ours)**	$\mathbf{2.69_{\pm 0.59}}$	38.5%	$\mathbf{5.10_{\pm 1.99}}$	62.7%	$\mathbf{0.998_{\pm 9.48E-04}}$
	R	$4.61_{\pm 1.29}$		$18.00_{\pm 9.63}$		$0.992_{\pm 4.49E-03}$
GPE-25*	**C (ours)**	$\mathbf{3.24_{\pm 0.06}}$	4.6%	$\mathbf{4.36_{\pm 0.17}}$	5.7%	$0.998_{\pm 8.7E-05}$
	R	$3.42_{\pm 0.19}$		$4.79_{\pm 0.64}$		$0.998_{\pm 3.22E-04}$
GPE-16*	**C (ours)**	$\mathbf{1.28_{\pm 0.03}}$	47.3%	$\mathbf{0.82_{\pm 0.04}}$	63.7%	$\mathbf{1.000_{\pm 1.79E-05}}$
	R	$3.18_{\pm 0.91}$		$6.74_{\pm 2.92}$		$0.997_{\pm 1.39E-03}$
GPE-36*	**C (ours)**	$\mathbf{7.13_{\pm 0.98}}$	7.1%	$\mathbf{20.90_{\pm 5.70}}$	9.6%	$\mathbf{0.989_{\pm 2.96E-03}}$
	R	$7.70_{\pm 0.85}$		$23.50_{\pm 5.31}$		$0.988_{\pm 2.81E-03}$
GPE-17	**C (ours)**	$\mathbf{4.07_{\pm 0.20}}$	2.7%	$\mathbf{5.21_{\pm 0.64}}$	7.8%	$0.999_{\pm 1.69E-04}$
	R	$4.18_{\pm 0.11}$		$5.61_{\pm 0.30}$		$0.999_{\pm 8.09E-05}$
GPE-27	**C (ours)**	$\mathbf{7.97_{\pm 0.39}}$	5.3%	$\mathbf{17.80_{\pm 2.52}}$	8.9%	$\mathbf{0.995_{\pm 7.68E-04}}$
	R	$8.42_{\pm 0.30}$		$19.60_{\pm 0.92}$		$0.994_{\pm 2.77E-04}$
Average	**C (ours)**	**3.57**	22.4%	**6.77**	32.6%	**0.997**
	R	6.36		65.30		0.976
Median	**C(ours)**	**3.16**	9.05%	**4.73**	14.7%	**0.998**
	R	4.40		8.69		0.995
Maximum	**C (ours)**	**7.13**	76.6%	**20.9**	94.3%	**0.999**
	R	31.5		826		0.999
Minimum	**C (ours)**	**0.31**	−1.4%	0.03	0.2%	**0.989**
	R	0.33		0.03		0.971

C:CV-PINN; R: PINN;
Each set of experiments was repeated three times.
Experiments marked with * use the PDM-sampling method.

the CV-PINN is more stable than the PINN. For instance, in solving GPE-13, the CV-PINN consistently maintained low errors across five repeated experiments with different random seeds, whereas the PINN exhibited a higher standard deviation in its errors.

In Fig. 6, we illustrate the differences between CV-PINN and PINN in predicting GPEs with prominent peaks and troughs, using GPE-34, GPE-16, and GPE-23 as examples. Both models generate predictions that closely match the basic shape of the reference solutions. However, the solutions provided by CV-PINN appear more "high-definition," containing greater detail, especially evident in the case of GPE-16 [Fig. 6(b)]. This enhanced clarity in CV-PINN's predictions is attributable to its more accurate peak predictions, whereas the PINN model tends to underpredict peak values and overpredict trough values, resulting in an image that appears "soft focused." A longitudinal comparison across the three cases reveals that such discrepancies at peak values become increasingly pronounced with the enhancement of solution nonlinearity, leading to larger errors in the PINN model under

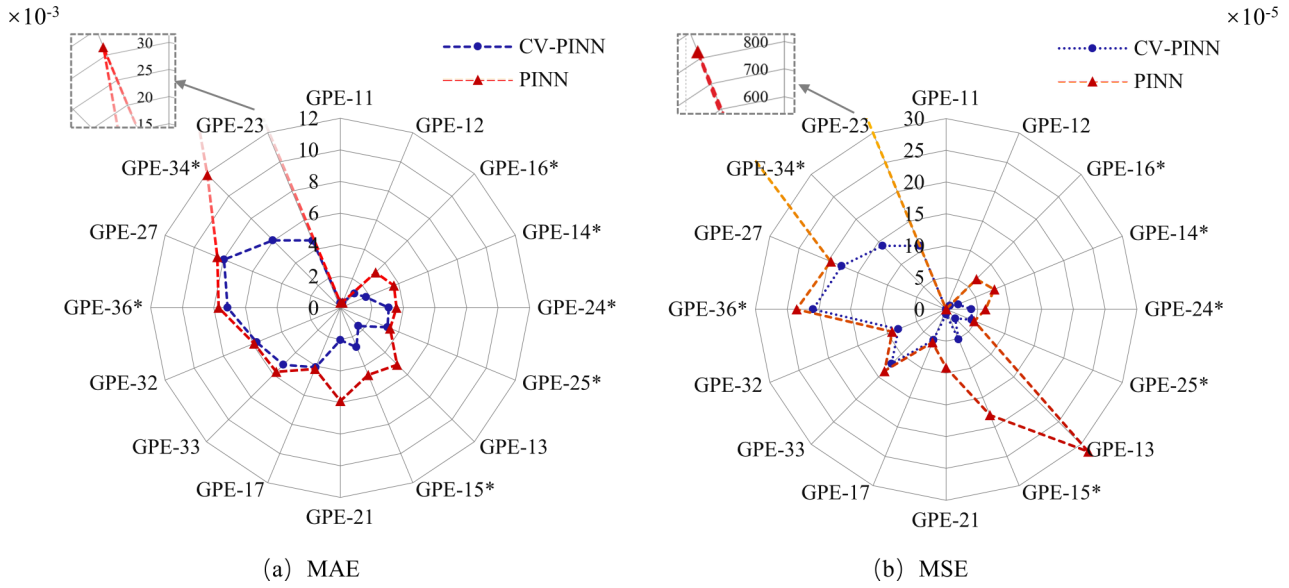

FIG. 5. Performance comparison of CV-PINN and PINN across various GPEs based on MAE and MSE metrics. The left chart (a) shows the MAE values, and the right chart (b) displays the MSE values. Overall, the polygon enclosed by the CV-PINN data points covers a smaller area on the radar chart, suggesting lower prediction errors across the 16 cases examined. Conversely, the PINN models not only cover larger polygon areas—indicative of higher errors—but also show substantial errors in specific cases. This characteristic is particularly pronounced in (b) MSE. Experiments marked with * use the PDM-sampling method.

conditions of more intense fluctuations. In contrast, CV-PINN consistently captures the high points of peaks, yielding more precise predictions.

B. Convergence performance

In addition to achieving higher prediction accuracy, CV-PINN also demonstrates significantly improved convergence and optimization speed compared to PINN. Figure 7 presents the loss decline curves and the corresponding test absolute error curves observed during the training process for four equations: GPE-11, GPE-13, GPE-23, and GPE-34. The left plots 7(a), 7(c), 7(e), and 7(g) illustrate that for the more complex GPE equations involving potential functions, such as GPE-13, GPE-23, and GPE-24, the loss decline curves for CV-PINN are steeper, indicating faster convergence. The right plots 7(b), 7(d), 7(f), and 7(h) further demonstrate that the MAE decline on the test set for CV-PINN is steeper and more rapid compared to PINN, indicating faster convergence and lower final MAE levels. Additionally, the range of standard errors in both plots shows that CV-PINN exhibits higher stability.

We also visualized a training process (GPE-14) for the two models, as illustrated in Fig. 8. This visualization illustrates the predicted outputs at 30 000, 50 000, 70 000, 90 000, and 100 000 iterations during the training process, providing insight into how the model evolves over time. It is evident from the figure that the CV-PINN model reached a relatively accurate solution after 50 000 iterations, with subsequent iterations further refining the solution towards the reference. In contrast, the training trajectory of the PINN model showed a significant lag, with notably slower adaptation. This is clearly reflected in the loss curves, where PINN approached convergence after approximately 40 000 iterations, but maintained a consistently higher error relative to CV-PINN, ultimately converging to a less accurate solution. Noteworthy is the observation from the MSE change curve in the lower left of the graph, indicating a minimal disparity between the two models. Numerical analysis reveals that both models achieved an R^2 exceeding 0.99, with MAE values of 2.18×10^{-3} for CV-PINN and 5.01×10^{-3} for PINN, pointing to minimal difference. However, the graphical depiction clearly illustrates substantial differences in predictive accuracy, with the PINN model exhibiting markedly evident errors. This reinforces the imperative for advancing more precise PINN models, as even slight inaccuracies can lead to considerable deviations between the predicted solutions and the true solutions.

Regarding the optimization speed of CV-PINN compared to PINN, as discussed earlier, CV-PINN achieves the same level of accuracy in fewer training epochs. This is attributed to its faster convergence properties. Despite this advantage, the forward and backward computational complexity of both models remains comparable, as the number of parameters in CV-PINN is similar to that in PINN. Additionally, the complex arithmetic embedded within CV-PINN is implemented through specialized matrix representations, ensuring computational efficiency. Consequently, the training time required for an equivalent number of iterations is nearly identical for both models. However, due to its faster convergence rate, CV-PINN reaches the desired accuracy significantly quicker, demonstrating superior time efficiency compared to PINN.

C. Parameter efficiency

The size of the model directly impacts its ability to adapt to problems of varying complexity. Larger models, with more parameters and more complex network structures, generally exhibit superior fitting and handling capabilities across a wide range of problems, from simple to complex. The previous

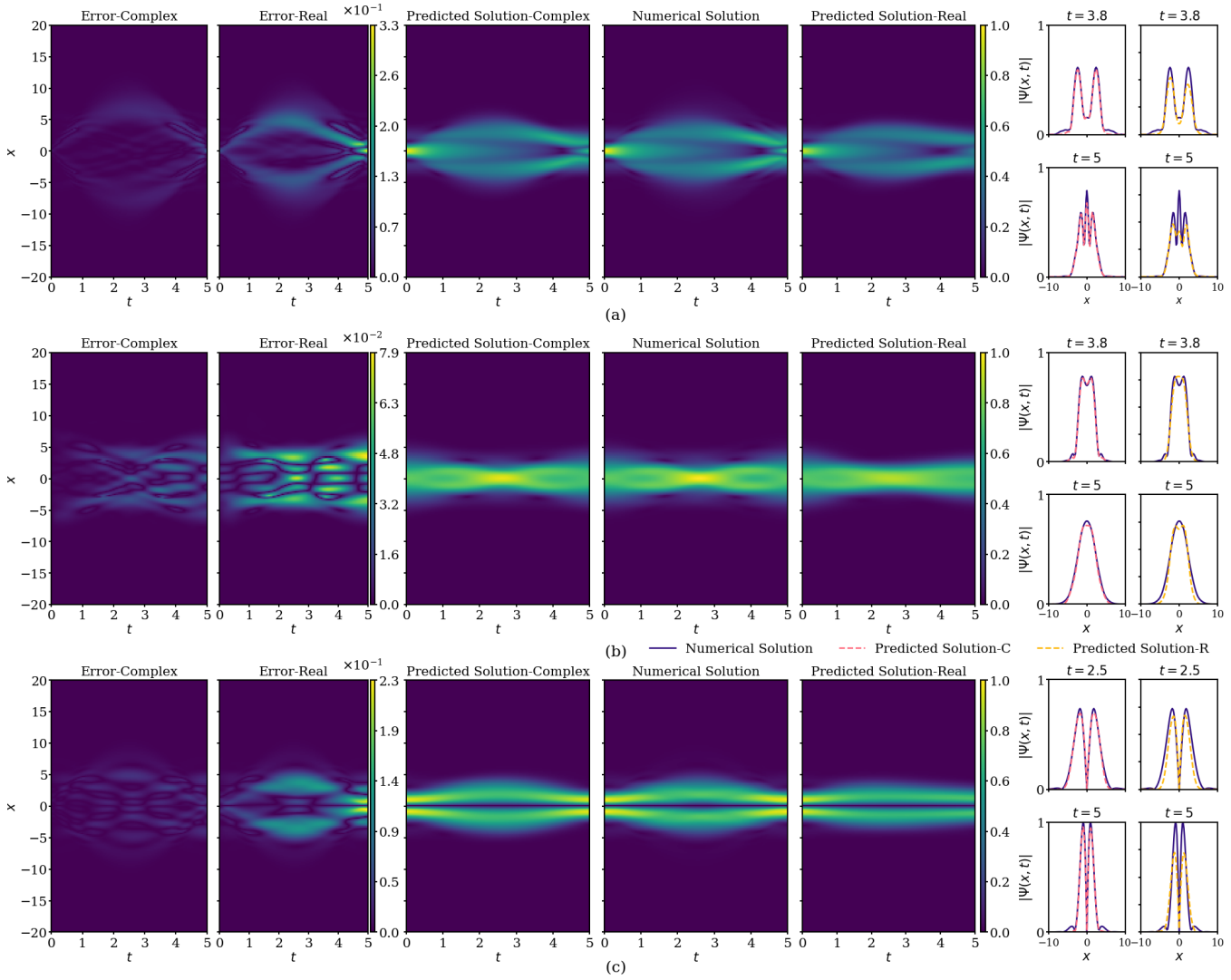

FIG. 6. Comparative analysis of prediction results. The figure presents the predictive outcomes of the CV-PINN and PINN for three GPE: GPE-34 (a), GPE-16 (b), and GPE-23 (c). Each column, from left to right, displays the prediction error of the CV-PINN model, the prediction error of the PINN model, the predictive results of the CV-PINN, the reference solutions, the predictive results of the PINN, and comparative curves of the solutions by CV-PINN and PINN at specific times.

experiments employed deeper and wider networks to ensure generalizability across all case equations. However, in scenarios where computational resources are limited, model size may need to be constrained. In this section, we reduce the number of layers and neurons to evaluate the parameter efficiency of CV-PINN and PINN using GPE-13 as an example.

As shown in Table IV, reducing the number of layers significantly impacts model performance. This effect is observed in both CV-PINN and RV-PINN, where fewer layers lead to a substantial increase in prediction errors. Notably, CV-PINN demonstrates superior performance and stability, with significantly lower average prediction error and standard deviation compared to RV-PINN. On the other hand, keeping the number of layers constant while reducing the number of neurons per layer has a milder effect on model performance. Although errors increase, both CV-PINN and PINN still provide relatively good prediction results. Comparatively, CV-PINN consistently outperforms PINN across the three sets of experiments. These five sets of experiments collectively demonstrate that CV-PINN exhibits greater robustness and parameter efficiency.

D. Effectiveness of the PDM-sampling method

As previously discussed, for some equations with strong nonlinearity or more complex representations, training models using uniform collocation-point sampling methods can result in model degradation or convergence to suboptimal solutions. For example, in certain equations like GPE-24, the model trained with conventional sampling methods failed to provide effective solutions even after 70 000 iterations. Figure 9 shows the training loss decline curves and evaluation error decline curves for GPE-24 and GPE-34 under both sampling methods. It can be observed that conventional methods exhibit high convergence points early in the training, where the model tends to degrade, yielding trivial solutions (globally zero). Even if the model escapes this degradation trend, it converges at a higher error.

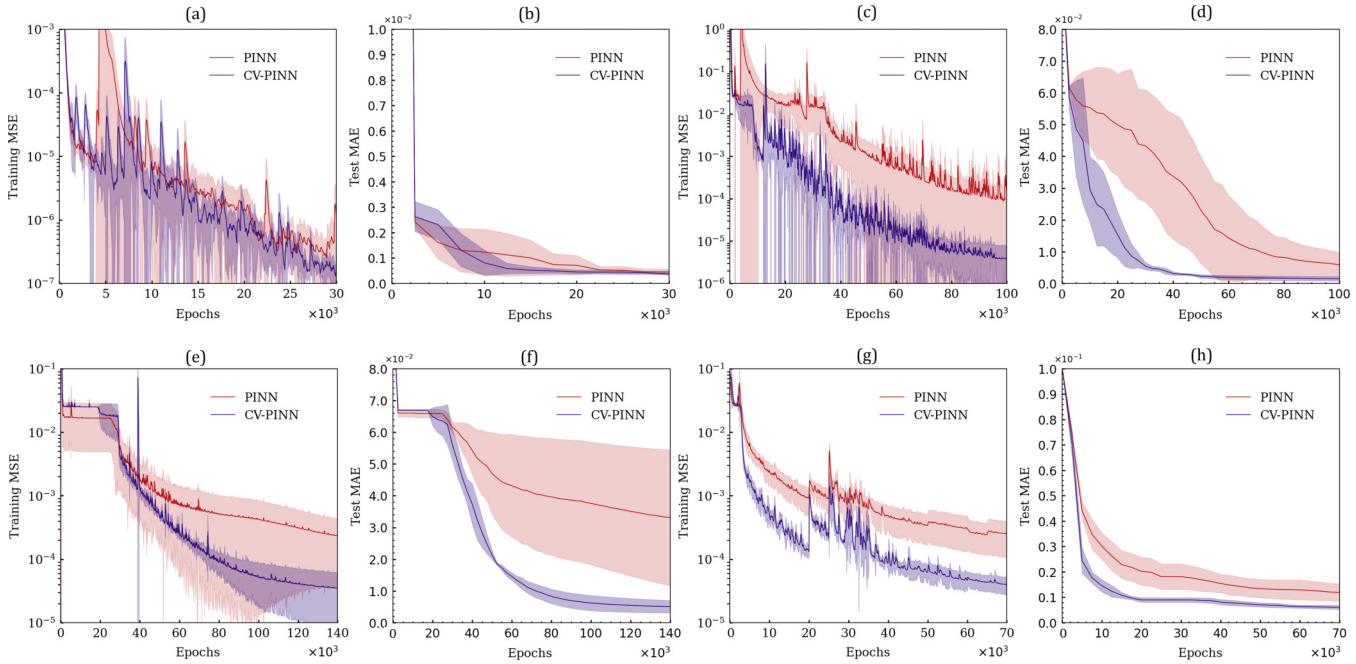

FIG. 7. Comparison of convergence performance between CV-PINN and PINN. This figure illustrates the training MSE decline curves and evaluation MAE decline curves for CV-PINN and PINN across four GPE equations. Specifically, plots (a), (c), (e), and (g) represent the training MSE decline curves for equations GPE-11, GPE-13, GPE-23, and GPE-34, respectively. Plots (b), (d), (f), and (h) depict the evaluation MAE decline curves for the same set of equations. To improve the visualization of the loss curve, exponential weighted moving average (EWMA) smoothing was applied.

For the evolution of time-dependent systems, initial conditions play a crucial role. The rapid decline in total loss during the PINN training process is synchronized with the decline in the initial condition term. In some failed training examples, such as GPE-24 and GPE-34 discussed in this section, the initial condition term loss barely decreases over an extended period. Although the PDE loss term decreases rapidly during this time, the model converges to trivial solutions, particularly when the solution itself is largely zero, as illustrated in Fig. 10(b).

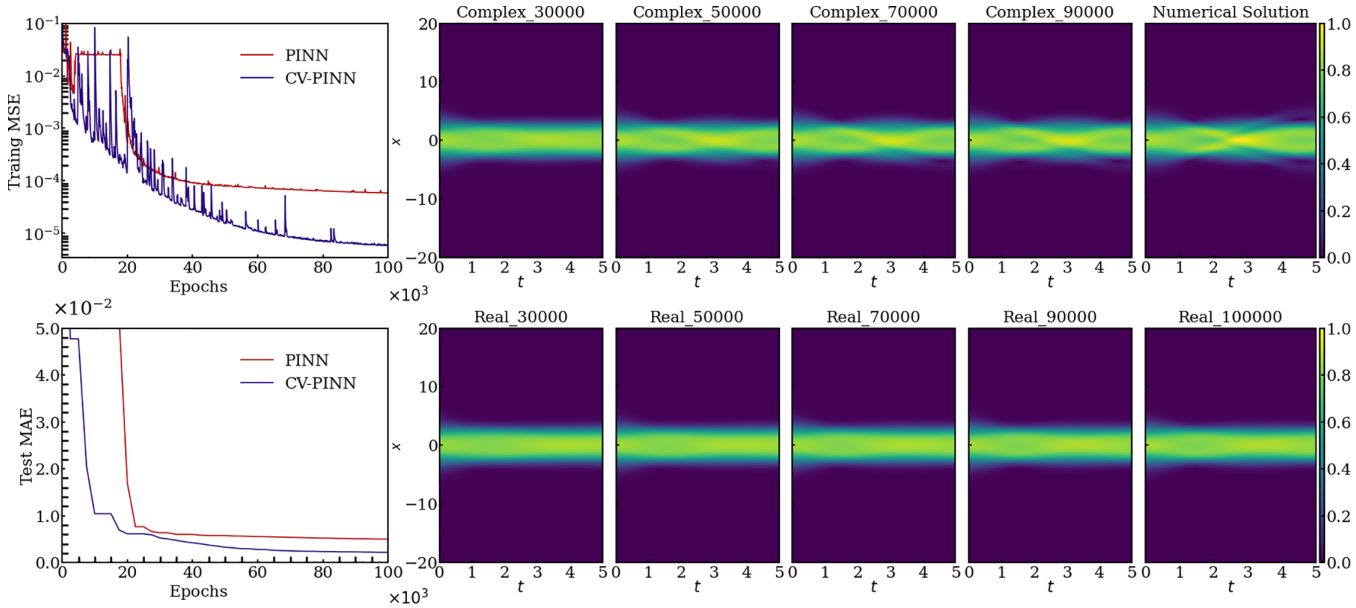

FIG. 8. Comparative training process visualization. This figure uses GPE-14 as an example to illustrate the training processes of the CV-PINN and PINN models. The two displayed loss curves represent the training loss decay curve (upper) and the testing error decay curve (lower), respectively. The series of images on the right show the reference solutions along with the predictions given by both models at specific checkpoints; "Complex_50000" denotes the solution provided by CV-PINN after completing 50 000 iterations, while "Real_50000" represents the prediction offered by the PINN model at the corresponding checkpoint, and so forth.

TABLE IV. Comparison of robustness between CV-PINN and PINN. The table compares the robustness of CV-PINN and PINN models under different settings with GPE-03. Better results are highlighted in bold.

Exp	Model	Setting	Trainable parameters	MAE ($\times 10^{-3}$)	DR_{MAE}	MSE ($\times 10^{-5}$)
1	**C (ours)**	Removing 1 layer from the base model	**10 602**	$3.73_{\pm 0.10}$	63.8%	$\mathbf{9.58_{\pm 0.52}}$
	R		10 649	$10.31_{\pm 1.63}$		$79.52_{\pm 25.22}$
2	**C (ours)**	Removing 2 layer from the base model	**5 502**	$18.06_{\pm 7.72}$	57.1%	$\mathbf{267.15_{\pm 223.73}}$
	R		5 468	$42.19_{\pm 12.03}$		$1417.36_{\pm 691.27}$
3	**C (ours)**	Reducing layer size	**10 162**	$1.66_{\pm 0.43}$	31.1%	$\mathbf{1.87_{\pm 1.03}}$
	R		10 205	$2.41_{\pm 0.29}$		$3.92_{\pm 0.88}$
4	**C (ours)**	Reducing layer size	**5 098**	$2.25_{\pm 0.65}$	42.7%	$\mathbf{3.56_{\pm 2.09}}$
	R		5 036	$3.93_{\pm 0.80}$		$10.93_{\pm 4.54}$
5	**C (ours)**	Reducing layer size	**2 682**	$2.69_{\pm 0.75}$	22.9%	$\mathbf{5.11_{\pm 3.11}}$
	R		2 578	$3.49_{\pm 0.72}$		$8.79_{\pm 4.18}$

C: CV-PINN; R: PINN; Each set of experiments was repeated three times.

The PDM-sampling method addresses this issue by using initial conditions to determine the density of collocation points during the preheating phase, temporarily restricting the solution domain to nonzero regions of the solution, effectively preventing model degradation. Subsequently, by dynamically adapting to the training progress, the collocation points can adaptively concentrate or diffuse into other regions, enabling training over the entire solution domain. As shown in Fig. 11, although collocation points initially concentrate in the middle of the solution domain during the warming phase, they quickly spread throughout the domain as training progresses.

Furthermore, since the PDM sampling can effectively concentrate a large number of collocation points in specific subregions of the solution domain, it significantly reduces the difficulty of model fitting. Therefore, even in cases that are less prone to degradation, the PDM sampling exhibits significant advantages over conventional sampling methods: faster convergence, higher optimization speed, and lower convergence error. We conducted a quantitative comparison using seven solving cases, and the results indicate that the PDM sampling achieves lower prediction errors and more accurate solutions when solving such PDEs, reducing the MSE by an average of 70.7%. Additionally, it enhances stability (see Fig. 12 and Table V).

It is important to note that although we introduced PDM sampling during the training process, this does not incur additional computational overhead. The required operations for monitoring and resampling primarily involve basic computations such as subtraction, addition, and averaging, all of which have a computational complexity of $\mathcal{O}(N)$, where N is the number of collocation points. These operations are computationally negligible compared to the $\mathcal{O}(N \cdot P)$ complexity of gradient backpropagation, where P represents the number of trainable network parameters. Compared to the computational cost of gradient backpropagation, the additional time required for resampling and evaluation is minimal and can be considered negligible. Our experimental observations confirm that under identical experimental settings, the runtime per backward pass for both models shows virtually no difference.

Moreover, unlike other training strategies that employ segmented or regionwise sequential training, our method maintains a global optimization approach over the entire solution domain, merely shifting the focus without introducing extra error accumulation or extending the training

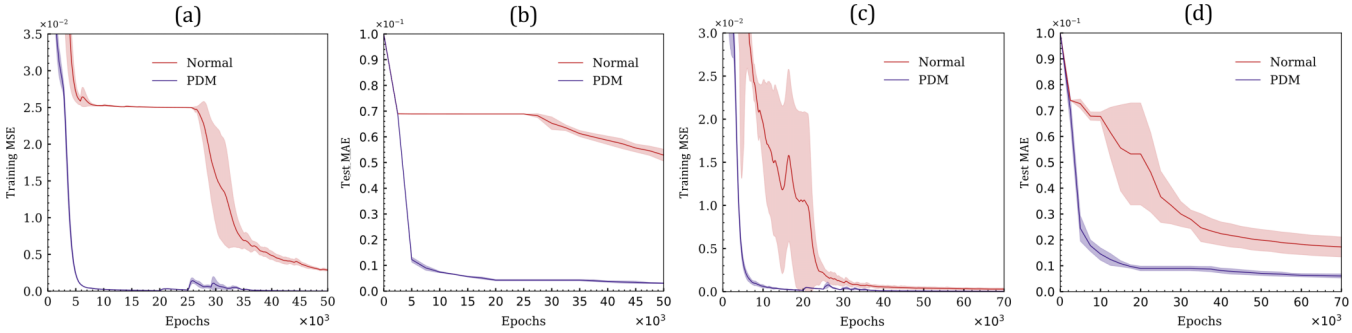

FIG. 9. Comparison of convergence performance under different sampling methods. This figure presents the training MSE decline curves and evaluation MAE decline curves for GPE-24 and GPE-34 using two different sampling methods: normal and PDM sampling. Specifically, plots (a) and (c) show the training MSE trends for GPE-24 and GPE-34, respectively. Plots (b) and (d) depict the evaluation MAE trends for the same equations.

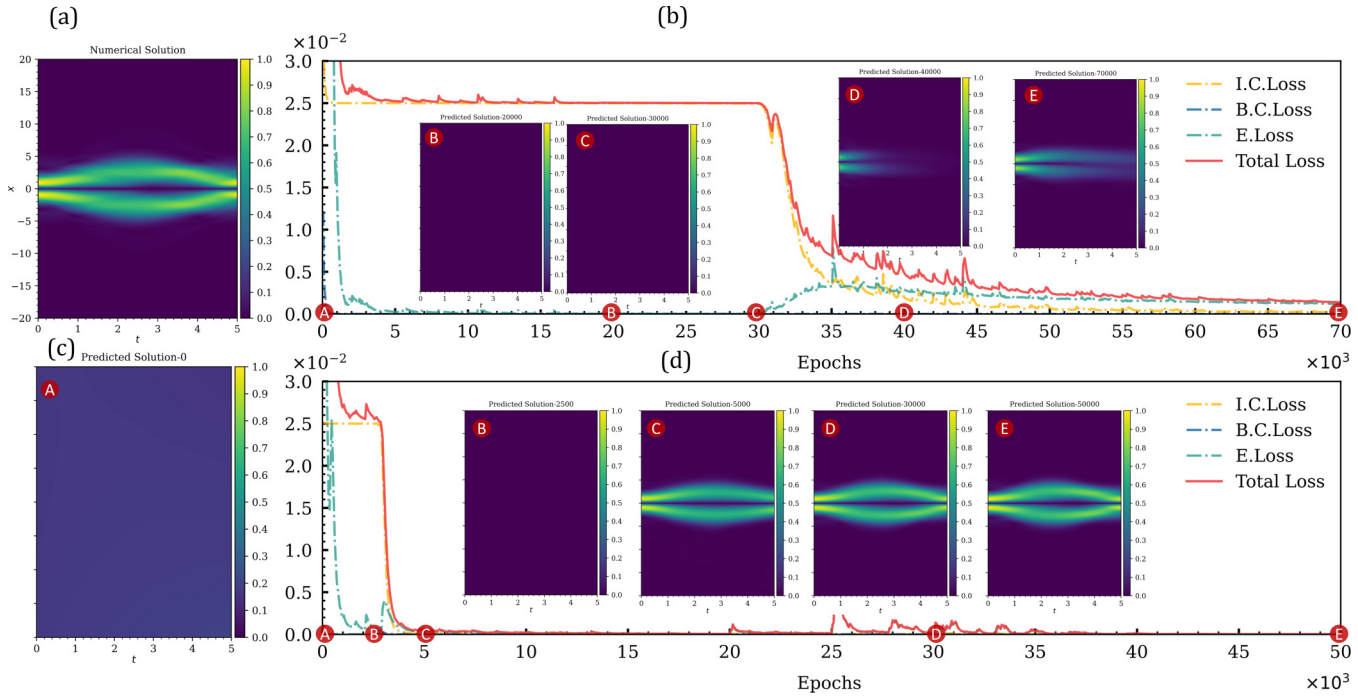

FIG. 10. Loss decline curves and corresponding solutions during training (GPE-24). The left panel (a) shows the reference solution for GPE-24, while (c) presents the solution given by the model after initialization, corresponding to training time A. The right panel (b) displays the training curves using the conventional sampling method, and (d) shows the training curves using PDM sampling. Subplots B to E in both figures represent the solutions provided by each model at corresponding training times. Apart from the difference in collocation-point sampling methods, all other settings for the two models are identical.

time. Additionally, since our method eliminates reliance on gradient-based resampling and incorporates a warm-up training phase, it effectively mitigates the model degradation issues commonly encountered by PINN models when solving complex systems—as previously discussed. This approach also avoids the need for additional gradient calculations for redistributing collocation points, further enhancing training efficiency.

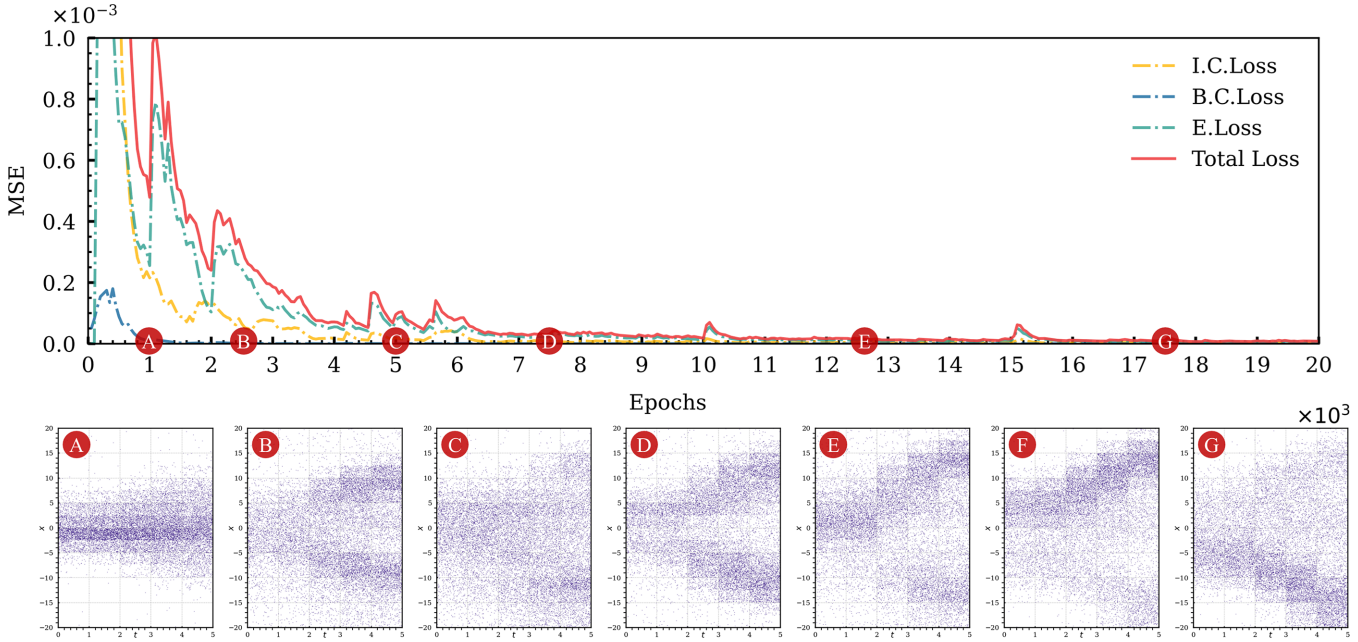

FIG. 11. Adaptive collocation-point distribution in PDM sampling. This figure illustrates the dynamic adaptation of collocation points during the training process using the PDM-sampling method. The top plot shows the decline in various loss components over epochs. The letters A to G correspond to specific stages in the training process, marked by arrows. The bottom subplots (A to G) depict the distribution of collocation points at these stages. Due to space constraints, only a subset of the collocation-point distributions is shown in the figure.

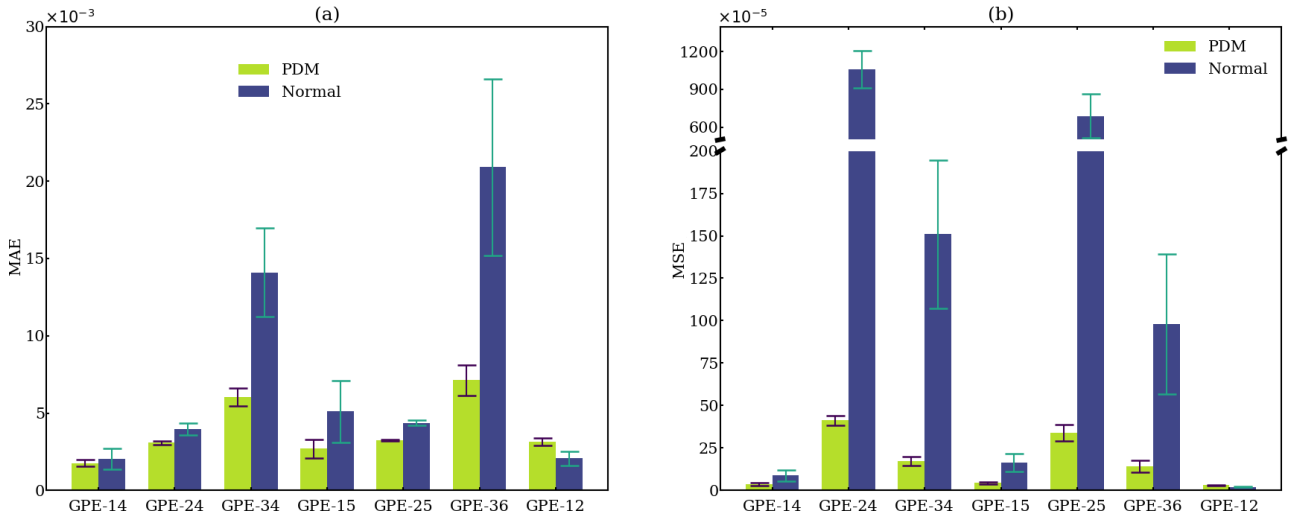

FIG. 12. Bar-chart comparison of PDM-sampling method and conventional sampling method in CV-PINN. (a) compares the MAE of the two methods, while (b) contrasts their MSE. It is evident from the figures that our proposed method exhibits significant advantages in both accuracy and stability, as indicated by lower bars and shorter error lines.

VI. CONCLUSION

To summarize, this paper introduced a complex-valued physics-informed neural network (CV-PINN) for solving quintic nonlinear Schrödinger equations. The CV-PINN framework integrates complex number representation and computation directly into the network structure, leveraging its enhanced representational capacity and ability to capture phase and amplitude information, making it particularly suitable for handling complex problems. Additionally, we proposed a collocation-point sampling method (PDM sampling), leveraging predictive dynamic analysis to address issues encountered during the training process. The PDM-sampling method innovatively determines the sampling density in various regions based on the dynamic model training behaviors. This approach prioritizes training in regions with significant

TABLE V. Effectiveness of PDM-sampling method vs conventional sampling method in CV-PINN. The table compares the performance of CV-PINN models using PDM-sampling method and conventional sampling method across different equations. Better results are highlighted in bold.

Equation	Model	MAE ($\times 10^{-3}$)	DR_{MAE}	MSE ($\times 10^{-5}$)	DR_{MSE}	R^2
GPE-14	**P (ours)**	**$1.77_{\pm 0.23}$**	47.5%	**$2.05_{\pm 0.67}$**	71.5%	**$0.999_{\pm 3.21E-04}$**
	N	$3.61_{\pm 0.78}$		$8.70_{\pm 3.13}$		$0.996_{\pm 1.52E-03}$
GPE-24	**P (ours)**	**$3.08_{\pm 0.12}$**	92.5%	**$3.97_{\pm 0.39}$**	99.6%	**$0.998_{\pm 1.96E-04}$**
	N	$41.30_{\pm 2.85}$		$1060.00_{\pm 146.00}$		$0.402_{\pm 4.40E-02}$
GPE-34	**P (ours)**	**$6.05_{\pm 0.57}$**	63.2%	**$14.10_{\pm 2.87}$**	88.7%	**$0.993_{\pm 1.50E-03}$**
	N	$17.30_{\pm 2.73}$		$151.00_{\pm 43.70}$		$0.912_{\pm 2.67E-02}$
GPE-15	**P (ours)**	**$2.69_{\pm 0.59}$**	40.3%	**$5.10_{\pm 1.99}$**	68.7%	**$0.998_{\pm 9.48E-04}$**
	N	$4.47_{\pm 0.64}$		$16.30_{\pm 5.28}$		$0.992_{\pm 2.53E-03}$
GPE-25	**P (ours)**	**$3.24_{\pm 0.06}$**	89.9%	**$4.36_{\pm 0.17}$**	99.2%	**$0.998_{\pm 8.70E-05}$**
	N	$33.90_{\pm 4.74}$		$688.00_{\pm 175.00}$		$0.572_{\pm 1.02E-01}$
GPE-36	**P (ours)**	**$7.13_{\pm 0.98}$**	45.7%	**$20.90_{\pm 5.70}$**	71.8%	**$0.989_{\pm 2.96E-03}$**
	N	$14.20_{\pm 3.53}$		$98.00_{\pm 41.40}$		$0.946_{\pm 2.35E-02}$
GPE-12	**P (ours)**	$3.16_{\pm 0.25}$	−3.6%	$2.07_{\pm 0.46}$	−4.5%	$0.999_{\pm 2.80E-04}$
	N	**$3.07_{\pm 0.27}$**		**$1.99_{\pm 0.31}$**		**$0.999_{\pm 1.94E-04}$**
Average	**P (ours)**	**3.87**	53.6%	**7.51**	70.7%	**0.996**
	N	16.84		289.14		0.831
Median	**P (ours)**	**3.16**	47.5%	**4.36**	71.8%	**0.997**
	N	14.20		98.00		0.946
Maximum	**P (ours)**	**7.13**	92.5%	**20.90**	99.6%	**0.999**
	N	41.30		1060.00		0.999
Minimum	**P (ours)**	**1.77**	−3.6%	2.05	−4.5%	**0.989**
	N	3.03		**1.99**		0.402

P: PDM sampling; N: Normal sampling; Each set of experiments was repeated three times.

solution variations, thereby accelerating convergence and reducing errors. We systematically evaluated the effectiveness of the proposed framework and method across 16 different GPEs, each with varying forms and initial conditions. The results indicate that compared to traditional real-valued PINN, complex-valued PINN exhibits significant advantages in convergence rate, error minimization, stability, and robustness, reducing the squared error by 32.6% and demonstrating 43.5% greater stability and robustness across multiple test cases. These advantages are especially pronounced when solving complex problems. The PDM-sampling method proves to be a simple yet highly effective approach that can effectively prevent model degradation and significantly accelerate convergence, with some cases showing particularly notable improvements, reducing MSE by an average of 70.7% compared to traditional methods. It is important to emphasize that while our primary focus was on solving GPEs, the CV-PINN framework can be readily applied to other types of complex-valued PDEs, and even to real-valued PDEs. Similarly, the PDM-sampling method is applicable to any PINN problem.

Limitations and future work. Currently, we have only considered the solution of one-dimensional GPEs. In the future, we plan to extend our approach to two-dimensional and three-dimensional problems and attempt to solve other types of complex-valued partial differential equations. It is worth noting that CV-PINN should not be limited to complex problems; previous studies have successfully applied quaternion neural network structures to real-valued problems [49]. Extending CV-PINN to the real domain holds great potential, and this is one of the directions for our future work. Regarding the PDM-sampling method, our study has only conducted preliminary experiments in this paper, where we simply monitored the changes in model predictions and uniformly partitioned the solution domain. In fact, there are many aspects of this method that can be optimized, such as simultaneously monitoring the equation residuals and solution gradients, and adaptively evaluating the training nodes that require resampling. These also can be pursued as part of future studies.

ACKNOWLEDGMENTS

This work was supported and partially funded by the National Natural Science Foundation of China (Grant No. 62106116), the National Center for Applied Mathematics Shenzhen (NCAMS), the Shenzhen Key Laboratory of Natural Gas Hydrates (Grant No. ZDSYS20200421111201738), the SUSTech – Qingdao New Energy Technology Research Institute, the China Meteorological Administration Climate Change Special Program (CMA-CCSP) (Grant No. QBZ202316), and the High Performance Computing Centers at Eastern Institute of Technology, Ningbo, and Ningbo Institute of Digital Twin.

DATA AVAILABILITY

The implementation details of the whole process and relevant data are available on GitHub [50].

[1] W. Bao and Y. Cai, Mathematical theory and numerical methods for Bose-Einstein condensation, Kinet. Relat. Models **6**, 1 (2013).

[2] P. Visscher, A fast explicit algorithm for the time-dependent Schrödinger equation, Comput. Phys. **5**, 596 (1991).

[3] A. Sacchetti, Spectral splitting method for nonlinear Schrödinger equations with quadratic potential, J. Comput. Phys. **459**, 111154 (2022).

[4] M. Thalhammer, M. Caliari, and C. Neuhauser, High-order time-splitting Hermite and Fourier spectral methods, J. Comput. Phys. **228**, 822 (2009).

[5] V. E. Zakharov and A. A. Gelash, Nonlinear stage of modulation instability, Phys. Rev. Lett. **111**, 054101 (2013).

[6] E. Small, An analysis of physics-informed neural networks, arXiv:2303.02890.

[7] M. Du, Y. Chen, and D. Zhang, AutoKE: An automatic knowledge embedding framework for scientific machine learning, IEEE Trans. Artif. Intell. **4**, 1564 (2022).

[8] F. Tori and V. Ginis, Phase space approach to solving higher order differential equations with artificial neural networks, Phys. Rev. Res. **4**, 043090 (2022).

[9] 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).

[10] A. D. Jagtap, K. Kawaguchi, and G. Em Karniadakis, Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, Proc. R. Soc. A **476**, 20200334 (2020).

[11] J. Pu and Y. Chen, Complex dynamics on the one-dimensional quantum droplets via time piecewise PINNs, Phys. D: Nonlinear Phenom. **454**, 133851 (2023).

[12] Y. Chen, D. Huang, D. Zhang, J. Zeng, N. Wang, H. Zhang, and J. Yan, Theory-guided Hard Constraint Projection (HCP): A knowledge-based data-driven scientific machine learning method, J. Comput. Phys. **445**, 110624 (2021).

[13] D. Zhang, L. Guo, and G. E. Karniadakis, Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks, SIAM J. Sci. Comput. **42**, A639 (2020).

[14] G. Pang, L. Lu, and G. E. Karniadakis, fPINNs: Fractional physics-informed neural networks, Siam J. Sci. Comput. **41**, A2603 (2019).

[15] S. Markidis, The old and the new: Can physics-informed deep-learning replace traditional linear solvers? Front. Big Data **4**, 669097 (2021).

[16] N. Wang, H. Chang, and D. Zhang, Deep-learning-based inverse modeling approaches: A subsurface flow example, J. Geophys. Res.: Solid Earth **126**, e2020JB020549 (2021).

[17] N. Wang, H. Chang, and D. Zhang, Surrogate and inverse modeling for two-phase flow in porous media via theory-guided convolutional neural network, J. Comput. Phys. **466**, 111419 (2022).

[18] X. Jiang, D. Wang, Q. Fan, M. Zhang, C. Lu, and A. Lau, Solving the nonlinear Schrödinger equation in optical fibers using

physics-informed neural network, in *Proceedings of the Optical Fiber Communication Conference (OFC)* (Optica Publishing Group, Washington, DC, USA, 2021), pp. 1–3.

[19] D. Liu, W. Zhang, Y. Gao, D. Fan, B. Malomed, and L. Zhang, Physics-informed neural network for nonlinear dynamics of self-trapped necklace beams, Opt. Express **32**, 38531 (2024).

[20] M. Zhong, S. Gong, S.-F. Tian, and Z. Yan, Data-driven rogue waves and parameters discovery in nearly integrable P T -symmetric Gross–Pitaevskii equations via PINNs deep learning, Physica D **439**, 133430 (2022).

[21] S. Zhang, P. Lan, and J.-J. Su, Wave-packet behaviors of the defocusing nonlinear Schrödinger equation based on the modified physics-informed neural networks, Chaos **31**, 113107 (2021).

[22] C. Wu, M. Zhu, Q. Tan, L. Lu, and M. Zhu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, Comput. Methods Appl. Mech. Eng. **403**, 115671 (2023).

[23] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Meth. Appl. Mech. Eng. **360**, 112789 (2020).

[24] G. K. R. Lau, A. Hemachandra, S.-K. Ng, and B. K. H. Low, *PINNACLE: PINN adaptive colocation and experimental points selection. The Twelfth International Conference on Learning Representations* (ICLR, Vienna, 2024).

[25] L. Li, W. Qiu, C. Dai, and Y. Wang, Prediction of self-similar waves in tapered graded index diffraction decreasing waveguide by the A-gPINN method, Nonlinear Dyn. **112**, 10319 (2024).

[26] J.-C. Pu, J. Li, and Y. Chen, Soliton, breather, and rogue wave solutions for solving the nonlinear Schrdinger equation using a deep learning method with physical constraints, Chinese Physics B **30**, 060202 (2021).

[27] H. Zhang, M. Gu, X. D. Jiang *et al.*, An optical neural chip for implementing complex-valued neural network, Nat. Commun. **12**, 457 (2021).

[28] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, Deep Complex Networks, in 6th International Conference on Learning Representations (ICLR, Vancouver, BC, Canada, 2018).

[29] M. Ko, U. K. Panchal, H. Andrade-Loarca, and A. Mendez-Vazquez, CoShNet: A Hybrid Complex Valued Neural Network using Shearlets arXiv:2208.06882.

[30] R. Abdalla, arXiv:2312.06087.

[31] Q. Wang, Z. Li, Z. Babic, W. Deng, L. Stanković, and D. P. Mandic, Widely linear matched filter: A lynchpin towards the interpretability of complex-valued CNNs, in 2024 International Joint Conference on Neural Networks (IJCNN) (Yokohama, Japan, 2024), pp. 1–8.

[32] K. Tiwari, N. M. Anoop Krishnan, and P. A. P, arXiv:2310.02094.

[33] Z. Mao and X. Meng, Physics-informed neural networks with residual/gradient-based adaptive sampling methods for solving partial differential equations with sharp solutions, Appl. Math. Mech.-Engl. Ed. **44**, 1069 (2023).

[34] J. Yu, L. Lu, X. Meng, and G. E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Meth. Appl. Mech. Eng. **393**, 114823 (2022).

[35] J. Florido, H. Wang, A. Khan, and P. K. Jimack, Investigating guiding information for adaptive collocation point sampling in PINNs, *Computational Science — ICCS 2024. Lecture Notes in Computer Science* (Springer, Cham, 2024), Vol. 14834.

[36] C. L. Wight and J. Zhao, Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks, Commun. Comput. Phys. **29**, 930 (2021).

[37] X. Cui, Quantum fluctuations on top of a PT-symmetric Bose-Einstein condensate, Phys. Rev. Res. **4**, 013047 (2022).

[38] C. Lee, H. Hasegawa, and S. Gao, Complex-valued neural networks: A comprehensive survey, IEEE/CAA J. Autom. Sin. **9**, 1406 (2022).

[39] H. Yu, Y. Liu, and M. Chen, Complex-valued neural-network-based federated learning for multiuser indoor positioning performance optimization, IEEE Internet of Things Journal **11**, 34065 (2024).

[40] J. W. Smith, arXiv:2309.07948.

[41] N. M. Müller, P. Sperl, and K. Böttinger, Complex-valued neural networks for voice anti-spoofing. INTERSPEECH 2023, 2831 August 2023, Dublin, Ireland (2023).

[42] Y. Kong, J. Wu, Q. Wang, P. Gao, W. Zhuang, Y. Wang, and L. Xie, Multi-channel automatic speech recognition using deep complex unet, in *2021 IEEE Spoken Language Technology Workshop (SLT)* (IEEE, Piscataway, NJ, 2021), pp. 104–110.

[43] C. Fang, Y. Song, F. Guan, F. Liang, and L. Yang, A robust complex-valued deep neural network for target recognition of UAV SAR imagery, IEEE J. Miniaturization Air Space Syst. **4**, 175 (2023).

[44] T. Nitta, An analysis of the fundamental structure of complex-valued neurons, Neural Process. Lett. **12**, 239 (2000).

[45] N. Mönning and S. Manandhar, Evaluation of complex-valued neural networks on real-valued classification tasks, arXiv:1811.12351.

[46] A. Hirose and S. Yoshida, Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence, IEEE Trans. Neural Networks Learn. Syst. **23**, 541 (2012).

[47] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. **43**, B1105 (2021).

[48] H. Xu, Y. Chen, and D. Zhang, Worth of prior knowledge for enhancing deep learning, Nexus **1**, 100003 (2024).

[49] Q. Cao, Y. Chen, D. Wang, Z. Xu, C. Ma, X. Yang, and S. Chen, Vision-informed flow field super-resolution with quaternion spatial modeling and dynamic fluid convolution, Phys. Fluids **36**, 095114 (2024).

[50] See https://github.com/ at Lei-Zhang1227/CV-PINN.