Stability in Training PINNs for Stiff PDEs: Why Initial Conditions Matter

Baoli Hao[1], Chun Liu[1], Ulisses Braga-Neto[2], Lifan Wang[3], and Ming Zhong[4,*]

[1]Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL 60616, USA
[2]Department of Electrical and Computing Engineering, Texas A&M University, College Station, TX 77843, USA
[3]Department of Physics and Astronomy, Texas A&M University, College Station, TX 77843, USA
[4]Department of Mathematics, University of Houston, Houston, TX 77204, USA

[*]Corresponding author: mzhong3@central.uh.edu

Abstract

Training physics-informed neural networks (PINNs) on stiff, time-dependent PDEs remains a fundamental challenge due to optimization instabilities and gradient pathologies. Through a series of rigorous ablation studies and Neural Tangent Kernel (NTK) analysis, we identify that the exact enforcement of initial conditions (ICs) is a decisive factor in stabilizing the training landscape. We present the first systematic ablation of two core strategies: hard initial-condition constrained transformation and self-adaptive loss weighting. Our findings demonstrate that embedding ICs directly into the network architecture provides an implicit time-marching effect, effectively reducing spectral bias and enabling the solution of highly stiff benchmarks, including sharp transitions and high-frequency coupled systems, primarily under periodic boundary conditions, with a Dirichlet extension reported as an additional robustness check. This work provides a scalable framework for developing reliable and physically-consistent neural solvers for complex mechanical systems.

Keywords. Physics-informed neural networks, stiff PDEs, hard initial-condition constraints, spectral bias, neural tangent kernel, optimization stability, stiff time integration.

Subject Classification. Primary: 65M12, 65M75; Secondary: 68T07, 65L04.

1 Introduction

Time-dependent partial differential equations (PDEs) constitute the mathematical foundation of numerous scientific and engineering disciplines, serving as essential tools for modeling a diverse range of physical phenomena. Their applications span from the intricate dynamics of fluid flow, as described by the Navier–Stokes equations, to the complex interplay of chemical reactions in biological systems, captured by reaction–diffusion models [1, 3, 5, 6, 9, 14, 17, 21, 22, 25, 26, 29, 31, 38, 39, 40, 44, 47, 48, 54, 57]. However, the solution of stiff time-dependent PDEs can be exceedingly challenging to obtain and often requires substantial computational resources and specialized numerical techniques, particularly due to multiscale effects and strong nonlinearities that are central to many problems in computational mechanics and engineering science.

Machine learning methodologies have demonstrated remarkable success across various scientific and engineering fields, transforming areas such as protein folding prediction [24], drug discovery and development [8], and even the optimization of fundamental algorithmic processes such as

matrix-vector multiplication [16]. However, traditional machine learning approaches, which rely on large datasets, can be impractical in scientific contexts where data acquisition is expensive, time-consuming, or simply infeasible. Moreover, purely data-driven models often lack physical fidelity and interpretability, as they may fail to capture the underlying physical principles governing the systems they aim to describe. Physics-informed machine learning (PIML), and in particular physics-informed neural networks (PINNs), address these limitations by integrating physical laws directly into the learning process. This significantly reduces the amount of training data required. By embedding constraints derived from governing PDEs into the neural network architecture, PINNs enable accurate predictions even when data are sparse or noisy [7, 11, 12, 23, 34, 41, 42, 43, 50, 56, 59]. Despite their promise, effectively training PINNs for stiff time-dependent PDEs remains a major challenge. The difficulty of propagating information from initial and boundary conditions into the computational domain, combined with the need to balance multiple loss terms, often results in convergence issues and suboptimal solutions [13, 37, 52, 53].

From a numerical perspective, this challenge is intimately connected to the ill-conditioning of the PDE solution operator with respect to initial and boundary data. Unlike traditional numerical methods, which exactly satisfy initial conditions at the start of the solution loop, PINNs enforce these conditions only in a soft, least-squares sense, making the training landscape highly sensitive to their degree of satisfaction. This ill-conditioning manifests as spectral bias in the optimization, causing PINNs to struggle with propagating information from initial conditions to subsequent time steps, which is a phenomenon widely documented in the literature [20, 28, 37, 51, 55].

This paper presents a complete and systematic study investigating the importance of training components for PINNs in solving stiff time-dependent PDEs. We do not claim the hard-constrained trial-function construction itself as novel: enforcing initial and/or boundary conditions exactly through an ansatz of the form $\tilde{u} = \psi + \phi\, u_{nn}$ dates back at least to [30], and has since been developed in various forms (see Section 1.1). Our contribution lies instead in how this construction is used and analyzed for stiff evolution problems. To the best of our knowledge, this is the first work to conduct a rigorous ablation study specifically isolating the role of exact initial-condition enforcement as the decisive stabilizing mechanism for stiff time-dependent forward problems, and to provide a theoretical justification for its necessity via Neural Tangent Kernel (NTK) analysis. We focus specifically on two training schemes: hard constraints and self-adaptive loss weights [37]. The hard-constraint scheme directly embeds initial and boundary conditions into the neural network architecture, reducing the multi-objective loss to a single-objective problem and mitigating spectral bias [53]. Through our ablation study, we conclude that the hard-constraint transformation is the most decisive factor in stabilizing training, effectively mimicking the behavior of traditional time integrators that begin from exact initial data. We further provide a rigorous NTK analysis demonstrating how this transformation simplifies the training dynamics and reduces spectral bias in a theoretically grounded manner. We emphasize that hard constraints are not the only effective strategy for handling stiff PDEs, and that this approach can be integrated with other advanced PINN techniques, such as causal PINNs [53], time-marching PINNs [55], residual-based-attention (RBA) PINNs [2], curriculum training [28], and co-training PINNs [58], to achieve further synergistic improvements in accuracy and robustness.

The rest of the paper is structured as follows. We compare our methodology to other related works in Section 1.1; in Section 2, we discuss the hard-constrained framework and other training methodologies used in our ablation and comparison study; we present a detailed discussion of the theoretical mechanism behind the hard-constrained transformation in Section 3; we present the results from five major categories of examples in Section 4; we conclude the paper in Section 5, pointing out several future directions for this line of research.

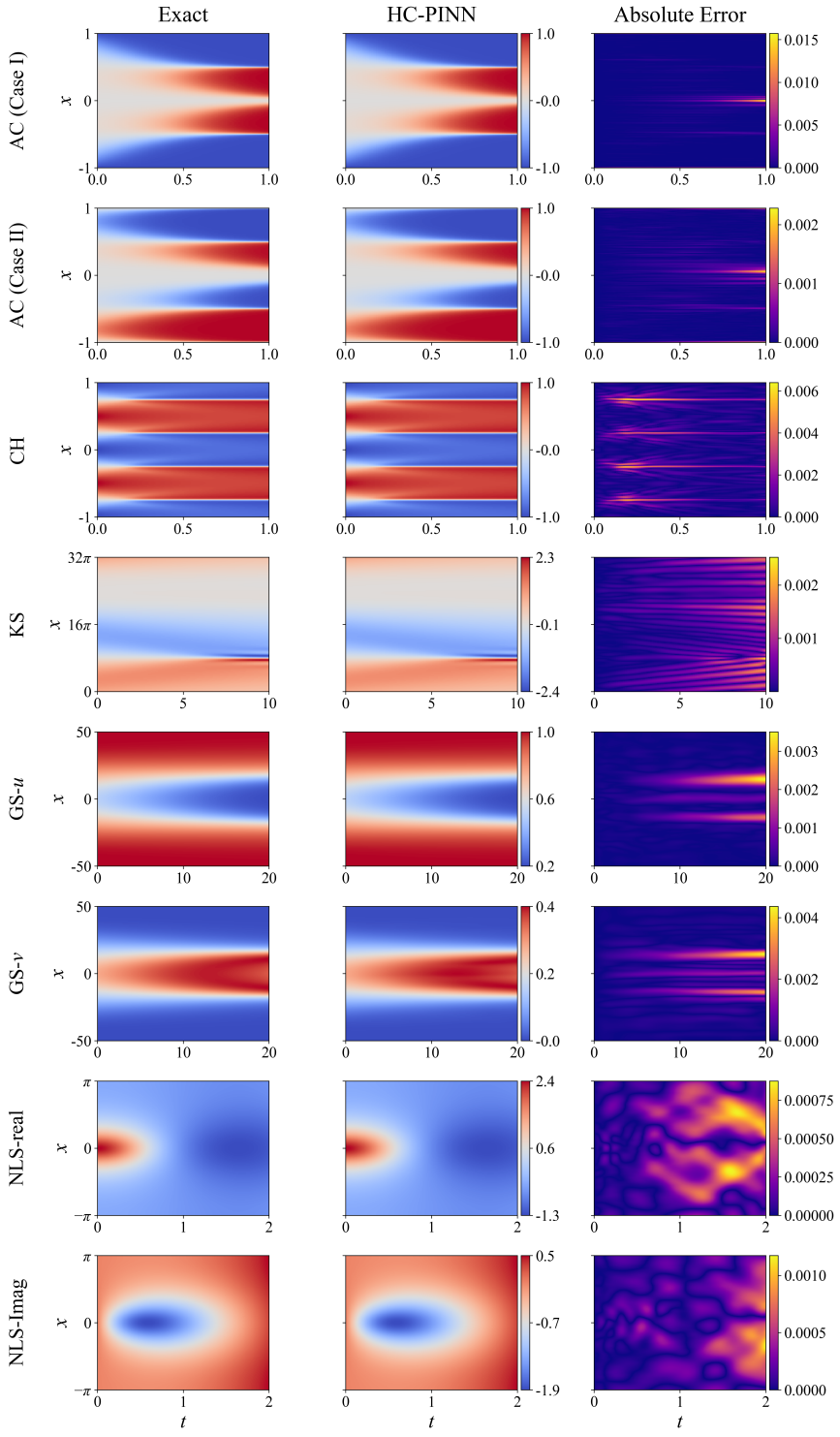

Figure 1: Benchmark Results: Truth vs HC-PINN vs Absolute Error.

1.1 Related work

PINNs have shown significant potential to solve a wide range of PDEs. However, challenges arise, particularly when dealing with "stiff" PDEs, where explicit numerical methods require extremely

small time steps for stability. A primary concern is the imbalance between the fitting of initial data and residual components within the PINN loss function [45, 52, 53, 55]. This imbalance often leads gradient descent to prioritize minimizing the residual loss over the data fitting loss, hindering convergence to accurate solutions. This issue is exacerbated in forward problems, where data is primarily concentrated at initial and boundary conditions, leaving limited information within the PDE domain. Specifically, for time-dependent problems, PINNs struggle to propagate information from initial conditions to subsequent time steps, a phenomenon widely documented in the literature [20, 28, 37, 51, 55]. Several strategies have been proposed to address these challenges. One approach, introduced by [55], involves time-marching, where the time domain is divided into smaller intervals, with PINNs being sequentially trained on each interval, starting from the initial condition. While effective, this method can be computationally expensive due to the need to train multiple networks. To improve information propagation, researchers have emphasized the importance of weighting initial condition data and residual points near the initial time [37, 55]. [37] demonstrated that their self-adaptive PINNs can autonomously learn these weights during training, particularly in scenarios with complex initial conditions, such as the wave equation. Further advancements include the "sequence-to-sequence" approach [28], causality PINN [51], artificial viscosity PINN [13], and co-training PINN [58]. Beyond loss weighting and time marching, structural modifications to PINNs have also been explored. [4] introduced Characteristic-Informed Neural Networks (CINNs) for transport equations, where PDE information is integrated into the neural network architecture. A similar approach, albeit for Dirichlet boundary conditions, was proposed in [30]. More recently, a growing body of work has developed hard-constraint PINN formulations along these architectural lines. [36] proposed hPINNs, which enforce constraints exactly through trial-function transformations and the penalty/ augmented-Lagrangian method, primarily in the context of inverse design and PDE-constrained optimization. [46] constructed approximate distance functions to enforce Dirichlet, Neumann, and Robin boundary conditions exactly, and related geometry-aware constructions have been used to impose exact boundary conditions in PINN and variational (VPINN) settings. Unified hard-constraint frameworks have also been proposed to handle multiple boundary-condition types within a single formulation [35]. These works demonstrate the value of exact constraint enforcement, but they are predominantly boundary-condition focused or aimed at inverse-design problems. They do not isolate exact initial-condition enforcement as the stabilizing mechanism for stiff time-dependent forward problems, nor provide an ablation- and NTK-based explanation of this effect. Our work addresses this gap by focusing specifically on exact initial-condition enforcement in stiff time-dependent forward problems.

2 Methodology

We consider the following setup for general time-dependent PDEs. Let u be an unknown function defined on $[0,T]\times\bar{\Omega}$ with $\Omega\subset\mathbb{R}^d$. Here the physical domain Ω comes with a Lipschitz boundary $\partial\Omega$ and $\bar{\Omega}=\Omega\cup\partial\Omega$. And we denote $\mathbf{x}=\begin{pmatrix}x_1 & \cdots & x_d\end{pmatrix}^\top\in\Omega$. Then we say that u is a solution of a time-dependent PDE, if u satisfies the following

$$\begin{cases} u_t(t,\mathbf{x})+\mathcal{P}[u](t,\mathbf{x}) &= f(t,\mathbf{x}), \quad (t,\mathbf{x})\in(0,T]\times\Omega,\\ u(0,\mathbf{x}) &= u_0(\mathbf{x}), \quad \mathbf{x}\in\bar{\Omega},\\ \mathcal{B}[u](t,\mathbf{x}) &= g(t,\mathbf{x}), \quad (t,\mathbf{x})\in[0,T]\times\partial\Omega, \end{cases} . \tag{1}$$

Here the operator $\mathcal{P}$ has order K and it is defined as

$$\mathcal{P}[u]=h(u,\{\partial^k_{x_1^{j_1}x_2^{j_2}\cdots x_d^{j_d}}u\}_{k=1}^K),\quad h \text{ is any multivariate function},$$

where $\partial^k_{x_1^{j_1}x_2^{j_2}\cdots x_d^{j_d}} u$ is a set that contains all of partial derivatives of u with respective to $\mathbf{x}$ of k^{th} order in the multivariate sense, i.e., for the powers of each coordinate, we have

$$j_1, j_2, \cdots, j_d \geq 0 \quad \text{and} \quad j_1 + j_2 + \cdots + j_d = k.$$

Furthermore, $\mathcal{B}$ denotes a boundary operator defined on $[0,T] \times \partial\Omega$. We assume that the operator $\mathcal{P}$, the boundary operator $\mathcal{B}$, and the prescribed data u_0, f, and g satisfy suitable conditions such that the corresponding initial-boundary value problem is well posed. We also assume that the initial and boundary data are compatible at $t = 0$, namely,

$$\mathcal{B}[u_0](\mathbf{x}) = g(0, \mathbf{x}), \qquad \mathbf{x} \in \partial\Omega.$$

For Dirichlet boundary conditions, this reduces to $u_0(\mathbf{x}) = g(0, \mathbf{x})$ on $\partial\Omega$.

Types of Boundary Conditions: one can consider many kinds of boundary conditions as follows

$$\mathcal{B}[u](t, \mathbf{x}; a, b) = au(t, \mathbf{x}) + b\frac{\partial u}{\partial \mathbf{n}}(t, \mathbf{x}), \quad \text{Robin type,}$$

where $\mathbf{n}$ is the outward normal vector to $\partial\Omega$ and a, b are two fixed constants. Special values of a and b can give two other BC cases, such as when $b = 0$ gives $\mathcal{B}[u](t, \mathbf{x}) = u(t, \mathbf{x})$ (Dirichlet) and $a = 0$ gives the $\mathcal{B}[u](t, \mathbf{x}) = \frac{\partial u}{\partial \mathbf{n}}(t, \mathbf{x})$ (Neumann). The periodic boundary condition gives d different equations, and they are,

$$\mathcal{B}[u](t, \mathbf{x}) = u(t, \mathbf{x}) - u(t, \mathbf{x} + P\mathbf{e}_i) = 0, \quad i = 1, \cdots, d. \tag{2}$$

Here P is the period and $\mathbf{e}_i$ is the i^{th} standard basis vector in $\mathbb{R}^d$, that is,

$$\mathbf{e}_i = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^\top,$$

where the entry 1 is in the i-th position. One can even consider a mixture of two or all of the above BC conditions. For example, we may decompose the boundary as

$$\partial\Omega = \Gamma_1 \cup \cdots \cup \Gamma_K,$$

where each Γ_i is associated with a different type of boundary condition. In this paper, we will mainly focus on the periodic type of boundary conditions.

Recent advancements in scientific machine learning integrate the principles of Physics, particularly the Physics-derived PDEs, into the training process of machine learning models. This integration enables the development of PINNs to solve for u as follows: find an approximate solution from a set of deep neural networks $\mathcal{H}_{\text{NN}}$, where each network has the same depth, the same number of neurons on each hidden layer, and the same activation functions on each layer, that minimizes the following loss functional

$$\text{Loss}(u_{nn}) = \text{Data Loss}(u_{nn}) + \lambda \cdot \underbrace{\text{IC Loss}(u_{nn}) + \text{BC Loss}(u_{nn}) + \text{PDE Loss}(u_{nn})}_{\text{Physical Loss}} \tag{3}$$

where λ is a regularization parameter, and the loss components are defined as follows

$$\begin{cases} \text{Data Loss}(u_{nn}) = \frac{1}{N_{Data}} \sum_{i=1}^{N_{Data}} |u_{nn}(t^{Data}, \mathbf{x}_i^{Data}) - u(t^{Data}, \mathbf{x}_i^{Data})|^2 \\ \text{IC Loss}(u_{nn}) = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} |u_{nn}(0, \mathbf{x}_i^{IC}) - u_0(\mathbf{x}_i^{IC})|^2, \\ \text{BC Loss}_{\text{per}}(u_{nn}) = \sum_{i=1}^{d} \frac{1}{N_{BC}} \sum_{j=1}^{N_{BC}} |u_{nn}(t_j^{BC}, \mathbf{x}_j^{BC}) - u_{nn}(t_j^{BC}, \mathbf{x}_j^{BC} + P\mathbf{e}_i)|^2, \\ \quad \text{or} \\ \text{BC Loss}_{\text{gen}}(u_{nn}) = \frac{1}{N_{BC}} \sum_{j=1}^{N_{BC}} |(\mathcal{B}[u_{nn}] - g)(t_j^{BC}, \mathbf{x}_j^{BC})|^2, \\ \text{PDE Loss}(u_{nn}) = \frac{1}{N_{CL}} \sum_{i=1}^{N_{CL}} |(\frac{\partial u_{\text{nn}}}{\partial t} + \mathcal{P}[u_{nn}] - f)(t_i^{CL}, \mathbf{x}_i^{CL})|^2 \end{cases}$$

for $u_{nn} \in \mathcal{H}_{\text{NN}}$. Note that BC $\text{Loss}_{\text{per}}(u_{nn})$ enforces continuity and periodicity along each spatial direction i, while in BC $\text{Loss}_{\text{gen}}(u_{nn})$, $\mathcal{B}$ and g denote the boundary operator (e.g., identity or normal derivative) and the prescribed boundary condition, respectively. Here $\{(t^{CL}, \mathbf{x}^{CL})_i\}_{i=1}^{N_{CL}} \in (0,T] \times \Omega$ are called collocation points, $\{(0, \mathbf{x}^{IC})_i\}_{i=1}^{N_{IC}} \in \{0\} \times \bar{\Omega}$ are the initial condition points, $\{(t^{BC}, \mathbf{x}^{BC})_i\}_{i=1}^{N_{BC}} \in [0,T] \times \partial\Omega$ are the boundary condition points, and $\{(t^{Data}, \mathbf{x}^{Data})_i\}_{i=1}^{N_{Data}} \in (0,T] \times \Omega$ are known data points (they can be noisy). The minimizer, denoted as u_{NN}, will be an approximate solution to (1).

2.1 Hard constraints

The main motivation for considering a hard-constraint transformation for PINNs is the possible ill-conditioning of the PDE solution operator's dependence on IC and BC. Traditional numerical methods do not encounter this kind of difficulty, as the IC data is exactly satisfied at the starting point of the solution loop. PINNs, on the other hand, use L^2-loss to fit the IC/BC and PDE residual data, which causes the solutions to be highly sensitive to the tightness of fit for the IC in the training, as pointed out in [37]. By transforming the need of fitting IC/BC conditions out of PINNs, it reduces the spectral bias in the training. Therefore, we consider the following transformation

$$\tilde{u}(t, \mathbf{x}) = \psi(t, \mathbf{x}) + \phi(t, \mathbf{x}) u_{nn}(t, \mathbf{x}), \tag{4}$$

Here ψ and ϕ are smooth functions chosen to enforce the initial condition exactly. In particular, we require

$$\psi(0, \mathbf{x}) = u_0(\mathbf{x}), \qquad \phi(0, \mathbf{x}) = 0, \qquad \mathbf{x} \in \bar{\Omega}. \tag{5}$$

For Dirichlet boundary conditions, the boundary condition can also be enforced exactly by requiring

$$\psi(t, \mathbf{x}) = g(t, \mathbf{x}), \qquad \phi(t, \mathbf{x}) = 0, \qquad (t, \mathbf{x}) \in [0,T] \times \partial\Omega. \tag{6}$$

For periodic boundary conditions, instead of (6), we assume that u_0, ψ, and ϕ are P-periodic in each spatial direction. Together with the periodic neural network construction introduced below, this guarantees that $\widetilde{u} = \psi + \phi u_{\text{nn}}$ satisfies the periodic boundary conditions exactly. When such transformation is used, then the training for u^{nn} is updated as

$$\text{Loss}_{HC}(u_{nn}) = \frac{1}{N_{CL}} \sum_{i=1}^{N_{CL}} |(\tilde{u}_t + \mathcal{P}[\tilde{u}] - f)(t_i^{CL}, \mathbf{x}_i^{CL})|^2, \quad \tilde{u} = \psi + \phi u_{nn}. \tag{7}$$

We have successfully reduced the multi-objective loss to a single-objective; however the information about IC and BC has been moved into the PDE residual loss through ψ and ϕ.

Periodic Neural Networks: even though we require ψ and ϕ to be periodic, we still need to build the information of periodicity into u_{nn}, hence we consider the following composition

$$u_{nn}(t, x) = f_{nn}(v(t, x)), \quad \text{where } f_{nn} \text{ is a neural network and } x \in \mathbb{R}.$$

Here $v(t, x) = \begin{bmatrix} t & \mathcal{F}_m(x) \end{bmatrix}^\top$ with

$$\mathcal{F}_m(x) = \begin{bmatrix} 1 & \cos\left(\frac{2\pi}{P}x\right) & \sin\left(\frac{2\pi}{P}x\right) & \cdots & \cos\left(\frac{2\pi}{P}mx\right) & \sin\left(\frac{2\pi}{P}mx\right) \end{bmatrix}$$

where m is a positive integer hyperparameter controlling the number of Fourier modes. With this construction, each trigonometric feature is exactly P–periodic, and therefore any linear combination (and hence u_{nn}) will inherit P–periodicity [15]. For $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}$, we can use the following

$$v(t, x, y) = \begin{bmatrix} t & \mathcal{F}_m(x) & \mathcal{F}_m(y) \end{bmatrix};$$

similarly for higher dimensional $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_d \end{bmatrix}$, we have

$$v(t, \mathbf{x}) = \begin{bmatrix} t & \mathcal{F}_m(x_1) & \cdots & \mathcal{F}_m(x_d) \end{bmatrix}.$$

2.2 Mini-batching

We have eliminated the IC/BC losses, but the transformed PDE loss might create additional difficulties in training due to the introduction of IC/BC functions into the PDE. We adopt a machine learning technique, called mini-batching, commonly used to reduce training complexity by evaluating the loss on randomly selected subsets of collocation data. In PINNs, this reduces memory costs, especially when handling large numbers of collocation points. Sampling mini-batches from collocation points also helps stabilize and accelerate optimization. So we consider mini-batching in the training for the PDE residual loss. As pointed out in [55], the mini-batching technique is similar to time-marching resampling but without an explicit time grid.

2.3 Self-adaptive PINNs

The proper choice of penalty term λ can be updated from derived formula as mentioned in [53]. However, data-driven discovery of λ is possible. Following [37], the loss is further modified as

$$\begin{aligned}
&\text{IC/BC Loss}(\{\lambda_i^{IC}\}_{i=1}^{N_{IC}}, \{\lambda_i^{BC}\}_{i=1}^{N_{BC}}) \\
&= \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} F_{mask}(\lambda_i^{IC}) |u_{nn}(0, \mathbf{x}_i^{IC}) - u_0(\mathbf{x}_i^{IC})|^2 \\
&\quad + \sum_{i=1}^{d} \frac{1}{N_{BC}} \sum_{j=1}^{N_{BC}} F_{mask}(\lambda_j^{BC}) |u_{nn}(t_j^{BC}, \mathbf{x}_j^{BC}) - u_{nn}(t_j^{BC}, \mathbf{x}_j^{BC} + P\mathbf{e}_i)|^2,
\end{aligned}$$

and

$$\text{Physical Loss}(\{\lambda_i^{CL}\}_{i=1}^{N_{CL}}) = \frac{1}{N_{CL}} \sum_{i=1}^{N_{CL}} F_{mask}(\lambda_i^{CL}) |(\frac{\partial u_{nn}}{\partial t} + \mathcal{P}(u_{nn}) - f)(t_i^{CL}, \mathbf{x}_i^{CL})|^2.$$

The set of loss weights, $\{\lambda_i^{IC}\}_{i=1}^{N_{IC}} \cup \{\lambda_i^{BC}\}_{i=1}^{N_{BC}} \cup \{\lambda_i^{CL}\}_{i=1}^{N_{CL}}$, are designed in a way to increase when needed during any training epoch in order to combat the spectral bias. Adaptive methods are essential to ensure that the neural network accurately addresses the challenging spots in the solutions of "stiff" PDEs. In our experiments, we employ self-adaptive PINNs as proposed by [37] to train PINNs adaptively. This approach involves fully trainable adaptation weights applied individually to each training point, allowing the neural network to independently identify and focus on the challenging regions of the solution.

2.4 Performance measures

Let $\{x_i, t_i\}_{i=1}^N$ be a collection of N data points where we evaluate both the reference solution $u(x,t)$ and the neural network output $\mathcal{U}(x,t)$. To measure the accuracy of the trained model, we compute the relative L^1 norm $\mathcal{E}_1$ and relative L^2 norm $\mathcal{E}_2$, defined by:

$$\mathcal{E}_1 = \frac{\sum_{i=1}^N |\mathcal{U}(x_i,t_i) - u(x_i,t_i)|}{\sum_{i=1}^N |u(x_i,t_i)|}, \quad \mathcal{E}_2 = \frac{\sqrt{\sum_{i=1}^N |\mathcal{U}(x_i,t_i) - u(x_i,t_i)|^2}}{\sqrt{\sum_{i=1}^N |u(x_i,t_i)|^2}}.$$

These metrics quantify the discrepancy between the predicted and reference solutions in the relative absolute-error (L^1) and quadratic/energy-error (L^2) senses, respectively. The relative normalization enables fair comparisons across different scales and variables. To rigorously assess the accuracy of

our results, we implemented a semi-implicit spectral method in Matlab. This allows us to make precise comparisons between the ground-truth solutions $u(x_i, t_i)$ and the PINN-generated solutions $\mathcal{U}(x_i, t_i)$.

3 Theoretical foundation for hard constraints

This section provides the theoretical justification for the stability and conditioning advantages of the hard-constraint transformation in PINNs. While the transformation enforces both boundary and initial conditions, our emphasis is on the initial condition, as the focus of this paper is on solving stiff time-dependent PDEs with PINNs. Stiff time-dependent PDEs exhibit features across widely varying time scales, making them difficult to capture even for explicit time-integration methods in traditional numerical approaches.

By incorporating the initial condition directly into the trial function, the hard-constraint transformation removes the initial-condition mismatch from the optimization problem. The network is therefore trained only through the transformed PDE residual, while every admissible approximation starts from the prescribed initial state. This is analogous to conventional time-dependent solvers, which evolve the solution from given initial data, and motivates the improved training stability observed below.

We divide the analysis into three parts: (I) consistency and non-degeneracy of the hard-constraint transformation, (II) Hessian and NTK characterization, and (III) implications for training dynamics, including decay and spectral bias.

3.1 Consistency and non-degeneracy of the hard-constraint transformation

We start with the minimal regularity conditions on the pair (ψ, ϕ) under which the hard-constraint transformation is consistent with the original initial-value problem and non-degenerate at the initial time.

Theorem 3.1 (Consistency and Non-Degeneracy of the Transformation)**.** *Assume that*

$$\phi, \psi \in C^1\big([0,T]; C^k(\bar{\Omega})\big),$$

and that

$$\psi(0,\mathbf{x}) = u_0(\mathbf{x}), \qquad \phi(0,\mathbf{x}) = 0, \qquad \phi_t(0,\mathbf{x}) \neq 0, \qquad \mathbf{x} \in \bar{\Omega}.$$

Let

$$u(t,\mathbf{x}) = \psi(t,\mathbf{x}) + \phi(t,\mathbf{x}) u_{nn}(t,\mathbf{x}).$$

Then u automatically satisfies the prescribed initial condition:

$$u(0,\mathbf{x}) = u_0(\mathbf{x}), \qquad \mathbf{x} \in \bar{\Omega}.$$

Moreover, u satisfies the original PDE

$$\partial_t u + \mathcal{P}[u] = f, \qquad (t,\mathbf{x}) \in (0,T] \times \Omega,$$

if and only if u_{nn} satisfies the transformed PDE

$$\phi \partial_t u_{nn} + \mathcal{P}[\psi + \phi u_{nn}] + \phi_t u_{nn} = f - \psi_t, \qquad (t,\mathbf{x}) \in (0,T] \times \Omega.$$

At the initial time, the value of the transformed variable is determined by

$$u_{nn}(0,\mathbf{x}) = \frac{u_t(0,\mathbf{x}) - \psi_t(0,\mathbf{x})}{\phi_t(0,\mathbf{x})}, \qquad \mathbf{x} \in \bar{\Omega}.$$

Therefore, the condition $\phi_t(0,\mathbf{x}) \neq 0$ prevents degeneracy in the recovery of $u_{nn}(0,\mathbf{x})$ from the transformed representation.

Remark 3.2. Theorem 3.1 shows that the hard-constraint transformation places the neural approximation on the correct initial-data manifold and provides a non-degenerate representation at $t = 0$. Intuitively, ϕ scales the learned correction while ψ encodes the prescribed initial condition. A bounded factor such as $1 - e^{-Ct}$ or t/T avoids the large-T growth associated with an unbounded choice such as $\phi = t$.

For example, one may choose

$$\psi(t,\mathbf{x}) = e^{-Ct}u_0(\mathbf{x}), \qquad \phi(t,\mathbf{x}) = 1 - e^{-Ct}.$$

In our implementation, we adopt the closely related choice

$$\psi(t,\mathbf{x}) = e^{-Ct}u_0(\mathbf{x}), \qquad \phi(t) = \frac{t}{T},$$

which is bounded on $[0,T]$ for any T and satisfies

$$\phi_t(0,\mathbf{x}) = \frac{1}{T} \neq 0.$$

This choice provides a simple normalized-time scaling while enforcing the initial condition exactly.

When $\mathcal{P}$ is linear and $\phi(t,\mathbf{x}) = \phi(t)$, the transformed PDE takes the simpler form

$$\phi\partial_t u_{nn} + \phi\mathcal{P}[u_{nn}] + \phi' u_{nn} = f - \psi_t - \mathcal{P}[\psi].$$

If $\phi(t) \neq 0$ for $t > 0$, then for positive times this equation can be written as

$$\partial_t u_{nn} + \mathcal{P}[u_{nn}] + \frac{\phi'}{\phi}u_{nn} = \frac{f - \psi_t - \mathcal{P}[\psi]}{\phi}.$$

Thus, u_{nn} satisfies an equation of the same general form as the original PDE, with an additional reaction term $\frac{\phi'}{\phi}u_{nn}$ and a modified forcing term that carries the information encoded in ψ, in particular the prescribed initial data.

3.2 NTK Perspective and training decay

Finally, we provide a Neural Tangent Kernel (NTK) perspective on how hard constraints would work to reduce the spectral bias and training difficulties of PINNs on solving stiff time-dependent PDEs. To simplify the discussion, we will focus on the PDEs with periodic boundary. Let $\{\mathbf{z}_i = (t_i,\mathbf{x}_i)\}_{i=1}^N \subset (0,T]\times\Omega$ denote collocation (residual) points and $\{\boldsymbol{\zeta}_j = (0,\mathbf{x}_j)\}_{j=1}^M$ denote initial-condition sample points. We define the two residual functions r_s and r_h as

$$r_s(\mathbf{z}_i) = \partial_t u_s(\mathbf{z}_i) + \mathcal{P}[u_s](\mathbf{z}_i) - f(\mathbf{z}_i),$$

and

$$r_h(\mathbf{z}_i) = \phi(\mathbf{z}_i)\partial_t u_h(\mathbf{z}_i) + \mathcal{P}[\psi + \phi u_h](\mathbf{z}_i) + \phi_t(\mathbf{z}_i)u_h(\mathbf{z}_i) - f(\mathbf{z}_i) + \psi_t(\mathbf{z}_i),$$

where $u_s = u_s(t, \mathbf{x}; \boldsymbol{\theta}_s)$ (soft constrained) and $u_h = u_h(t, \mathbf{x}; \boldsymbol{\theta}_h)$ (hard constrained) are the two PINN approximations, which are automatically periodic due to the Fourier layer. The corresponding losses to train u_s and u_h are

$$\text{Loss}_s(u_s(\boldsymbol{\theta}_s)) = \text{Loss}_{IC}(u_s(\boldsymbol{\theta}_s)) + \text{Loss}_{PDE}(u_s(\boldsymbol{\theta}_s))$$

with

$$\begin{cases} \text{Loss}_{IC}(u_s(\boldsymbol{\theta}_s)) & = \frac{1}{2M}\sum_{i=1}^{M} |u_s(\boldsymbol{\zeta}_i; \boldsymbol{\theta}_s) - u_0(\mathbf{x}_i^{IC})|^2, \quad \boldsymbol{\zeta}_i = (0, \mathbf{x}_i^{IC}), \\ \text{Loss}_{PDE}(u_s(\boldsymbol{\theta}_s)) & = \frac{1}{2N}\sum_{i=1}^{N} |r_s(\mathbf{z}_i; \boldsymbol{\theta}_s)|^2, \end{cases}$$

and $\text{Loss}_h(u_h(\boldsymbol{\theta}_h)) = \frac{1}{2N}\sum_{i=1}^{N} |r_h(\mathbf{z}_i; \boldsymbol{\theta}_h)|^2$.

Both Losses can induce a continuous-time (in terms of τ not t as t represents the physical time) gradient flow systems as

$$\frac{\mathrm{d}\boldsymbol{\theta}_s(\tau)}{\mathrm{d}\tau} = -\nabla \text{Loss}_s(u_s(\boldsymbol{\theta}_s(\tau))) \quad \text{and} \quad \frac{\mathrm{d}\boldsymbol{\theta}_h(\tau)}{\mathrm{d}\tau} = -\nabla \text{Loss}_h(u_h(\boldsymbol{\theta}_h(\tau))),$$

with $\boldsymbol{\theta}_s(0) = \boldsymbol{\theta}_h(0) = \boldsymbol{\theta}_0$. Let the operators $\mathcal{L}_s[u] \coloneqq \partial_t u + \mathcal{P}[u]$ and $\mathcal{L}_h[u] \coloneqq \phi\partial_t u + \mathcal{P}[\psi + \phi u] + \phi_t u$. For notational simplicity, in the following NTK dynamics we suppress the normalization factors arising from $1/M$ and $1/N$, since they only rescale the corresponding kernel contributions and do not affect the comparison between the coupled soft-constrained system and the residual-only hard-constrained system. Then with NTK, we obtain

$$\begin{bmatrix} \frac{\mathrm{d}u_s(\boldsymbol{\zeta};\boldsymbol{\theta}_s(\tau))}{\mathrm{d}\tau} \\ \frac{\mathrm{d}\mathcal{L}_s[u_s](\mathbf{z};\boldsymbol{\theta}_s(\tau))}{\mathrm{d}\tau} \end{bmatrix} = -\begin{bmatrix} \mathbf{K}_{11}^s(\tau) & \mathbf{K}_{12}^s(\tau) \\ \mathbf{K}_{21}^s(\tau) & \mathbf{K}_{22}^s(\tau) \end{bmatrix} \cdot \begin{bmatrix} u_s(\boldsymbol{\zeta}; \boldsymbol{\theta}_s(\tau)) - u_0(\mathbf{x}^{IC}) \\ \mathcal{L}_s[u_s](\mathbf{z}; \boldsymbol{\theta}_s(\tau)) - f(\mathbf{z}) \end{bmatrix}.$$

with $\mathbf{K}_{11}^s(\tau) \in \mathbb{R}^{M\times M}$, $\mathbf{K}_{12}(\tau) \in \mathbb{R}^{M\times N}$ ($\mathbf{K}_{12}^s(\tau) = (\mathbf{K}_{21}^s(\tau))^\top$), and $\mathbf{K}_{22}^s(\tau) \in \mathbb{R}^{N\times N}$ and the entries are given as

$$\begin{cases} (\mathbf{K}_{11}^s)_{i,j}(\tau) & = \langle \partial_{\boldsymbol{\theta}} u_s(\boldsymbol{\zeta}_i; \boldsymbol{\theta}_s(\tau)), \partial_{\boldsymbol{\theta}} u_s(\boldsymbol{\zeta}_j; \boldsymbol{\theta}_s(\tau)) \rangle \\ (\mathbf{K}_{12}^s)_{i,j}(\tau) & = \langle \partial_{\boldsymbol{\theta}} u_s(\boldsymbol{\zeta}_i; \boldsymbol{\theta}_s(\tau)), \partial_{\boldsymbol{\theta}} \mathcal{L}_s[u_s](\mathbf{z}_j; \boldsymbol{\theta}_s(\tau)) \rangle \\ (\mathbf{K}_{22}^s)_{i,j}(\tau) & = \langle \partial_{\boldsymbol{\theta}} \mathcal{L}_s[u_s](\mathbf{z}_i; \boldsymbol{\theta}_s(\tau)), \partial_{\boldsymbol{\theta}} \mathcal{L}_s[u_s](\mathbf{z}_j; \boldsymbol{\theta}_s(\tau)) \rangle \end{cases}.$$

Meanwhile, for u_h, we have

$$\frac{\mathrm{d}\mathcal{L}_h[u_h](\mathbf{z}; \boldsymbol{\theta}_h(\tau))}{\mathrm{d}\tau} = -\mathbf{K}^h(\tau) \left(\mathcal{L}_h[u_h](\mathbf{z}; \boldsymbol{\theta}_h(\tau)) - f(\mathbf{z}) + \psi_t(\mathbf{z})\right), \quad \mathbf{K}^h(\tau) \in \mathbb{R}^{N\times N},$$

with the entries of $\mathbf{K}^h$ being

$$(\mathbf{K}^h)_{i,j}(\tau) = \langle \partial_{\boldsymbol{\theta}} \mathcal{L}_h[u_h](\mathbf{z}_i; \boldsymbol{\theta}_h(\tau)), \partial_{\boldsymbol{\theta}} \mathcal{L}_h[u_h](\mathbf{z}_j; \boldsymbol{\theta}_h(\tau)) \rangle.$$

Setting

$$\mathbf{K}^s = \begin{bmatrix} \mathbf{K}_{11}^s & \mathbf{K}_{12}^s \\ \mathbf{K}_{21}^s & \mathbf{K}_{22}^s \end{bmatrix},$$

and assuming

$$\mathbf{K}^s(\tau) \approx \mathbf{K}^s(0) = \mathbf{K}_*^s, \qquad \mathbf{K}^h(\tau) \approx \mathbf{K}^h(0) = \mathbf{K}_*^h,$$

the soft-constrained residual satisfies

$$\begin{aligned}\begin{bmatrix} u_s(\boldsymbol{\zeta};\boldsymbol{\theta}_s(\tau)) - u_0(\mathbf{x}^{IC}) \\ \mathcal{L}_s[u_s](\mathbf{z};\boldsymbol{\theta}_s(\tau)) - f(\mathbf{z}) \end{bmatrix} &= e^{-\mathbf{K}_*^s \tau} \begin{bmatrix} u_s(\boldsymbol{\zeta};\boldsymbol{\theta}_s(0)) - u_0(\mathbf{x}^{IC}) \\ \mathcal{L}_s[u_s](\mathbf{z};\boldsymbol{\theta}_s(0)) - f(\mathbf{z}) \end{bmatrix} \\ &= \mathbf{Q}_s^\top e^{-\boldsymbol{\Lambda}_s \tau} \mathbf{Q}_s \begin{bmatrix} u_s(\boldsymbol{\zeta};\boldsymbol{\theta}_s(0)) - u_0(\mathbf{x}^{IC}) \\ \mathcal{L}_s[u_s](\mathbf{z};\boldsymbol{\theta}_s(0)) - f(\mathbf{z}) \end{bmatrix},\end{aligned}$$

where $\mathbf{K}_*^s = \mathbf{Q}_s^\top \boldsymbol{\Lambda}_s \mathbf{Q}_s$. Moreover, the hard-constrained residual satisfies

$$\begin{aligned}&\mathcal{L}_h[u_h](\mathbf{z};\boldsymbol{\theta}_h(\tau)) - f(\mathbf{z}) + \psi_t(\mathbf{z}) \\ &= e^{-\mathbf{K}_*^h \tau} \left(\mathcal{L}_h[u_h](\mathbf{z};\boldsymbol{\theta}_h(0)) - f(\mathbf{z}) + \psi_t(\mathbf{z}) \right) \\ &= \mathbf{Q}_h^\top e^{-\boldsymbol{\Lambda}_h \tau} \mathbf{Q}_h \left(\mathcal{L}_h[u_h](\mathbf{z};\boldsymbol{\theta}_h(0)) - f(\mathbf{z}) + \psi_t(\mathbf{z}) \right),\end{aligned}$$

where $\mathbf{K}_*^h = \mathbf{Q}_h^\top \boldsymbol{\Lambda}_h \mathbf{Q}_h$. One can see that, in the soft-constrained formulation, the training decay depends on the spectral properties of the full coupled block matrix $\mathbf{K}^s$, which contains both the initial-condition and PDE-residual components. In contrast, the training decay of the hard-constrained formulation depends only on the transformed PDE-residual kernel $\mathbf{K}^h$, since the initial-condition residual has been eliminated from the training system. This reduction helps mitigate the spectral imbalance between competing loss components.

Theorem 3.3 (Hessian Characterization and Bounds)**.** *Assume that $\mathcal{P}$ is linear and that $\phi(t, \mathbf{x}) = \phi(t) \neq 0$ for $t > 0$. Consider a linear-in-parameters representation of the network output, or equivalently its NTK linearization about a fixed parameter initialization,*

$$u(\mathbf{y};\boldsymbol{\theta}) = \mathbf{h}^\top(\mathbf{y})\boldsymbol{\theta}, \qquad \mathbf{h}(\mathbf{y}) = \begin{bmatrix} h_1(\mathbf{y}) & \cdots & h_D(\mathbf{y}) \end{bmatrix}^\top,$$

where $\mathbf{y} = \mathbf{z} = (t, \mathbf{x}) \in (0, T] \times \Omega$ or $\mathbf{y} = \boldsymbol{\zeta} = (0, \mathbf{x}^{IC}) \in \{0\} \times \bar{\Omega}$. Therefore the two losses become

$$\begin{aligned}&\mathit{Loss}_s(u(\mathbf{y};\boldsymbol{\theta})) = \frac{1}{2N} |\mathbf{A}\boldsymbol{\theta} - \mathbf{f}|^2_{\ell^2(\mathbb{R}^N)} + \frac{1}{2M} |\mathbf{C}\boldsymbol{\theta} - \mathbf{u}_0|^2_{\ell^2(\mathbb{R}^M)}, \ \textit{(soft constrained)} \\ &\mathit{Loss}_h(u(\mathbf{z};\boldsymbol{\theta})) = \frac{1}{2N} |\boldsymbol{\Lambda}_\phi \mathbf{A}\boldsymbol{\theta} + \boldsymbol{\Lambda}_{\phi_t} \mathbf{B}\boldsymbol{\theta} - \tilde{\mathbf{f}}|^2_{\ell^2(\mathbb{R}^N)}, \ \textit{(hard constrained)}\end{aligned}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathcal{L}_s[\mathbf{h}^\top(\mathbf{z}_1)] \\ \vdots \\ \mathcal{L}_s[\mathbf{h}^\top(\mathbf{z}_N)] \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{h}^\top(\mathbf{z}_1) \\ \vdots \\ \mathbf{h}^\top(\mathbf{z}_N) \end{bmatrix}, \quad \textit{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{h}^\top(\boldsymbol{\zeta}_1) \\ \vdots \\ \mathbf{h}^\top(\boldsymbol{\zeta}_M) \end{bmatrix},$$

where $\mathbf{x}_i^{IC} \in \bar{\Omega}$; and for the diagonal matrices $\boldsymbol{\Lambda}_\phi$ and $\boldsymbol{\Lambda}_{\phi_t}$

$$\boldsymbol{\Lambda}_\phi = \mathit{diag}(\phi(\mathbf{z}_1), \cdots, \phi(\mathbf{z}_N)) \quad \textit{and} \quad \boldsymbol{\Lambda}_{\phi_t} = \mathit{diag}(\phi_t(\mathbf{z}_1), \cdots, \phi_t(\mathbf{z}_N)),$$

and the vectors

$$\mathbf{f} = \begin{bmatrix} f(\mathbf{z}_1) \\ \vdots \\ f(\mathbf{z}_N) \end{bmatrix}, \quad \tilde{\mathbf{f}} = \begin{bmatrix} f(\mathbf{z}_1) - \mathcal{L}_s[\psi](\mathbf{z}_1) \\ \vdots \\ f(\mathbf{z}_N) - \mathcal{L}_s[\psi](\mathbf{z}_N) \end{bmatrix}, \quad \textit{and} \quad \mathbf{u}_0 = \begin{bmatrix} u_0(\mathbf{x}_1^{IC}) \\ \vdots \\ u_0(\mathbf{x}_M^{IC}) \end{bmatrix}.$$

Therefore the Hessians of the losses are

$$H_s(\boldsymbol{\theta}) = \frac{1}{N}\mathbf{A}^\top\mathbf{A} + \frac{1}{M}\mathbf{C}^\top\mathbf{C},\ H_h(\boldsymbol{\theta})$$
$$= \frac{1}{N}\Big(\mathbf{A}^\top\boldsymbol{\Lambda}_\phi^2\mathbf{A} + \mathbf{A}^\top\boldsymbol{\Lambda}_\phi\boldsymbol{\Lambda}_{\phi_t}\mathbf{B} + \mathbf{B}^\top\boldsymbol{\Lambda}_{\phi_t}\boldsymbol{\Lambda}_\phi\mathbf{A} + \mathbf{B}^\top\boldsymbol{\Lambda}_{\phi_t}^2\mathbf{B}\Big),$$

Then the Hessians are bounded as

$$|H_s|_2 \le \frac{|\mathcal{L}_s|_2^2}{N}|\mathbf{B}|_2^2 + \frac{1}{M}|\mathbf{C}|_2^2, \quad |H_h|_2 \le \frac{1}{N}(C_1|\mathcal{L}_s|_2^2 + 2C_2|\mathcal{L}_s|_2 + C_3)|\mathbf{B}|_2^2.$$

where $|\mathcal{L}_s|_2$ *denotes an induced discrete operator bound on the sampled linearized trial space such that*

$$|\mathbf{A}|_2 \le |\mathcal{L}_s|_2|\mathbf{B}|_2,$$

and

$$C_1 = \max_{\mathbf{z}} \phi^2(\mathbf{z}), \quad C_2 = \max_{\mathbf{z}} |\phi(\mathbf{z})\phi_t(\mathbf{z})|, \quad \textit{and} \quad C_3 = \max_{\mathbf{z}} \phi_t^2(\mathbf{z}).$$

Remark 3.4. The soft-constrained Hessian contains separate contributions from the PDE residual and the initial-condition mismatch, represented by $\mathbf{A}^\top\mathbf{A}$ and $\mathbf{C}^\top\mathbf{C}$, respectively. In contrast, the hard-constrained Hessian is determined entirely by the transformed PDE residual through $\boldsymbol{\Lambda}_\phi\mathbf{A} + \boldsymbol{\Lambda}_{\phi_t}\mathbf{B}$, since the initial-condition residual has been eliminated from the loss. This structural simplification is consistent with the NTK decay analysis above. When $\mathcal{P}$ is nonlinear or ϕ also depends on $\mathbf{x}$, additional terms arise and the corresponding Hessian characterization becomes more involved.

4 Examples

In this section, we examine the hard-constrained PINN (HC-PINN) on a diverse set of prototypical stiff, time-dependent PDEs with periodic boundary conditions, aiming to demonstrate its robustness in addressing a broad class of challenging problems. The benchmarks in Section 4.2 – 4.7 all use periodic boundary conditions. Section 4.10 additionally verifies that the hard-IC mechanism extends to non-periodic settings by reporting a Dirichlet Allen–Cahn run with a standard MLP (no Fourier features). These equations often involve strong nonlinearities, stiffness, and parameter sensitivities, which pose significant challenges for data-driven methods such as PINNs. Nonetheless, our results illustrate that the proposed framework can effectively overcome these difficulties and accurately approximate solutions across these regimes.

We present the results in five major categories: (I) an ablation study across 7 stiff PDEs with different sources of stiffness, including sharp phase transitions in Allen–Cahn, higher-order derivatives (fourth order) in Cahn–Hilliard and Kuramoto–Sivashinsky, coupled outputs in Gray–Scott and Schrödinger systems, and high-frequency modes in fast-moving linear transport; (II) comparisons with state-of-the-art PINNs, including the Causal PINN [51] and Enhanced RBA PINN [2], on Allen–Cahn, Cahn–Hilliard, and Kuramoto–Sivashinsky; (III) application of HC-PINNs to a $2D$ Allen–Cahn equation; (IV) sensitivity analysis with respect to the parameter m in the Allen–Cahn equation under two different initial conditions; and (V), as an additional robustness check beyond the periodic Fourier-feature setting, a non-periodic Dirichlet Allen–Cahn run with a plain MLP, reported in Section 4.10.

4.1 Implementation details

All experiments were implemented in PyTorch ($v2.2$) and executed on CUDA-enabled NVIDIA GPUs as well as Apple Silicon devices via the MPS backend. Our code runs efficiently across Linux (Ubuntu 22.04), macOS (Sonoma), and Windows 11 platforms. Primary training was conducted on an NVIDIA $A100$ GPU (40 GB memory) and a MacBook Pro with an Apple M1 Pro chip (16-core GPU, 32 GB unified memory).

For each benchmark, we used a fully connected neural network with 7 hidden layers, each containing 32 neurons and tanh activation functions, to represent the latent solution. The same initial weights were used across all ablation experiments for consistency.

Model training was carried out using a two-stage strategy: we first trained with the Adam optimizer using an initial learning rate of 10^{-3} for 50,000 steps, followed by fine-tuning with the L-BFGS-B optimizer with a learning rate of 1 for 5,000 steps. No hyperparameter sweeps were performed, and the total number of training iterations was fixed across all experiments.

This work does not rely on external datasets. We used Latin Hypercube Sampling (LHS) to generate a fixed set of 16,384 interior collocation points, along with 128 initial data points and 128 boundary data points. This sampling configuration was kept consistent across all experiments to ensure comparability. For PDE systems, we employed a mini-batching strategy with 5 batches of size 3,280 to improve memory efficiency during training.

To facilitate training and better incorporate the initial condition into the solution structure, we adopt the transformation

$$\tilde{u}(t,\mathbf{x}) = e^{-Ct}u_0(\mathbf{x}) + (t/T)\,u_{nn}(t,\mathbf{x}), \quad \psi = e^{-Ct}u_0(\mathbf{x}),$$
$$\phi(t) = t/T, \quad (t,\mathbf{x}) \in [0,T]\times\bar{\Omega},$$

where $u_0(\mathbf{x})$ denotes the initial condition. The factor $\phi(t) = t/T$ is equivalent to training the network on the normalized time coordinate $t/T \in [0,1]$, which is what we use in practice. This setting keeps ϕ bounded on the simulation interval for any final time T, while preserving the non-degeneracy $\phi_t(0,x) = 1/T \neq 0$ required by Theorem 3.1. We test two values of $C \in \{0.1, 1\}$ to assess how the decay rate affects the learned dynamics. The transformed PINN u_{nn} is trained by minimizing the PDE-residual loss, while the initial condition is enforced exactly by construction now.

Ablation studies were conducted by disabling one component of the method at a time while keeping all other settings fixed, to evaluate its contribution using relative L^2 and relative L^1 error metrics. These quantify quadratic/energy-error and absolute-error discrepancies, respectively, and are normalized to enable fair comparisons across different scales.

We also compare HC-PINN with causal training (MLP) and enhanced RBA using the same data and a total of 50,000 iterations, instead of the 300,000 iterations used in the original works. This choice is made deliberately to enforce an equal compute budget across all methods, so that differences in accuracy reflect the training schemes themselves rather than disparities in optimization effort; we acknowledge that this reduced budget may lower the performance of the baseline methods relative to their originally reported values, and we report those original 300,000-iteration values explicitly in Table 8 for transparency. For all baselines we use the recommended/default hyperparameters from their respective original works, and, consistent with the no-sweep protocol stated above, no additional tuning was performed for either the baselines or HC-PINN. For the sensitivity study, experiments with different m values are conducted on the same set of points, using the same number of iterations and architecture. For $2D$ training, we use 4,096 initial collocation points, 10,000 residual collocation points per segment, and 800 boundary points per segment. The architecture remains the same as in the $1D$ setting.

4.2 Allen–Cahn PDE

The Allen–Cahn type PDE is a classical phase-field model that has been widely used to study phase separation phenomena [3]. It has numerous practical applications across a range of fields, including materials science [1, 44], biological systems [22, 47], and electrochemical systems [21, 48]. We begin with the 1-dim Allen–Cahn equation with periodic boundary conditions formulated as follows:

$$\begin{aligned} &u_t - \gamma_1 u_{xx} + \gamma_2(u^3 - u) = 0,\ (t,x) \in (0,T] \times (a,b), \\ &u(0,x) = u_0(x),\ x \in [a,b], \quad u(t,a) = u(t,b),\ t \in [0,T], \end{aligned} \tag{8}$$

where $\gamma_1, \gamma_2 > 0, T > 0, a < b$ are prescribed constants. As γ_2 increases, the transition interface of the solutions is sharper, which makes it harder to solve the AC equation numerically.

4.2.1 Case I

We first tested on Allen–Cahn equation with the following, $u_0(x) = x^2\cos(\pi x)$, $T = 1$, $a = -1$, $b = 1$, $\gamma_1 = 0.001$, and $\gamma_2 = 5$.

For this example, we first test the baseline PINN approach [43]. As shown in Table 1, the baseline PINN alone fails to accurately solve the Allen–Cahn equation, motivating the development of a more effective training pipeline. To address this, we employ all proposed techniques (except mini-batching) and conduct an ablation study by disabling individual components one at a time. Additionally, we introduce two different decay functions for the initial condition, e^{-1t} and $e^{-0.1t}$, to investigate their influence on the solution. The results are summarized in Table 1. The full scheme, including self-adaptive loss balancing, achieves the best performance, reaching a relative L^2 error of 8.29×10^{-4} and relative L^1 error of 5.34×10^{-4}. In particular, using the faster-decaying initial condition ($IC1$) leads to better results than the slower-decaying case ($IC2$), suggesting that the initial condition plays a more critical role during the early stages of training but becomes less dominant over time. The ablation results further highlight the importance of each methodological component. Removing any single part generally leads to performance degradation, with the most severe impact observed when the initial condition is omitted, resulting in a relative error close to 0.99 in both L^1 and L^2 norms. Figure 2 presents cross-sectional comparisons at different times, further confirming the excellent agreement between the predicted and true solutions across the entire time horizon.

4.2.2 Case II

For the second case, we change the IC to have highly oscillatory data by $u(0,x) = x^2\sin(2\pi x)$. Other parameters are $T = 1$, $a = -1$, $b = 1$, $\gamma_1 = 0.001$ and $\gamma_2 = 4$.

We first train the baseline PINN model, which produces a large relative L^2 error of 0.511 and relative L^1 error of 0.328, demonstrating its inadequacy in resolving the sharp interface dynamics (Table 1). However, once the initial condition enforcement and periodic boundary neural networks are incorporated, the performance improves drastically, achieving a relative L^2 error of 8.32×10^{-4} and relative L^1 error of 3.72×10^{-4}. Adding self-adaptive weighting further improves the full-scheme errors to 3.53×10^{-4} and 2.14×10^{-4} (Figure 2, bottom). Further gains are realized when using the self-adaptive loss weighting strategy, although the improvement is marginal compared to the dominant contribution from enforcing the initial and boundary conditions. Interestingly, unlike the previous case, there is minimal performance difference between the schemes using $IC1$ and $IC2$, suggesting that the impact of initial condition decay becomes less significant when strong oscillations are already present at the beginning.

Ablation Settings				Case I		Case II	
PBC	**IC1**	**IC2**	**SA**	**Rel. L^2 error**	**Rel. L^1 error**	**Rel. L^2 error**	**Rel. L^1 error**
✓	✓	✗	✓	8.29×10^{-4}	5.34×10^{-4}	3.53×10^{-4}	2.14×10^{-4}
✓	✗	✓	✓	7.64×10^{-2}	1.66×10^{-2}	7.04×10^{-4}	3.26×10^{-4}
✓	✓	✗	✗	1.24×10^{-3}	6.10×10^{-4}	8.32×10^{-4}	3.72×10^{-4}
✓	✗	✓	✗	2.86×10^{-2}	8.20×10^{-3}	9.80×10^{-4}	3.91×10^{-4}
✓	✗	✗	✗	9.99×10^{-1}	9.99×10^{-1}	9.99×10^{-1}	9.99×10^{-1}
✓	✗	✗	✓	9.99×10^{-1}	9.99×10^{-1}	9.99×10^{-1}	9.99×10^{-1}
✗	✓	✗	✗	1.05×10^{-2}	2.65×10^{-3}	5.11×10^{-2}	1.21×10^{-2}
✗	✓	✗	✓	1.86×10^{-3}	7.31×10^{-4}	5.03×10^{-2}	8.27×10^{-3}
✗	✗	✓	✗	1.88×10^{-2}	4.71×10^{-3}	4.58×10^{-2}	1.10×10^{-2}
✗	✗	✓	✓	4.34×10^{-3}	1.24×10^{-3}	3.03×10^{-2}	4.25×10^{-3}
✗	✗	✗	✓	5.12×10^{-1}	3.18×10^{-1}	4.02×10^{-1}	1.42×10^{-1}
✗	✗	✗	✗	9.99×10^{-1}	9.98×10^{-1}	5.11×10^{-1}	3.28×10^{-1}

Table 1: *Allen–Cahn (Case I and Case II):* Relative L^2 and L^1 errors for an ablation study illustrating the impact of disabling individual components of the proposed technique and training pipeline.

4.3 Kuramoto–Sivashinsky PDE

Kuramoto–Sivashinsky (KS) equation is a fourth-order nonlinear partial differential equation, known for its chaotic behavior [29, 38]. It has served as a pivotal model in the study of complex dynamical phenomena observed in various physical systems. Consequently, the KS equation provides an excellent case study to demonstrate how PINNs can effectively simulate chaotic dynamics. The equation is given by:

$$\begin{aligned} u_t &= -u_{xx} - u_{xxxx} - uu_x, \quad (t,x) \in (0,T] \times (a,b), \\ u(0,x) &= \cos(\frac{x}{16})(1+\sin\frac{x-1}{16}), \quad x \in [a,b], \quad u(t,a) = u(t,b),\ t \in [0,T], \end{aligned} \tag{9}$$

where $T = 10, a = 0, b = 32\pi$. The best results are shown in Figure 3.

As in Table 2, solving this PDE system using a baseline PINN alone proves to be exceptionally challenging. The incorporation of periodic neural networks for boundary conditions, paired with initial condition *IC2*, consistently yields superior results compared to *IC1*. This suggests that, in

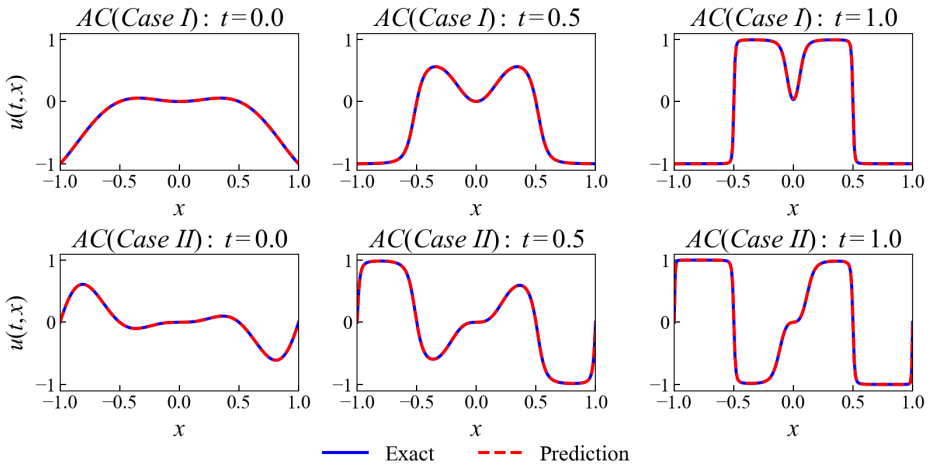

Figure 2: Top: Best solution of the *Allen–Cahn Equation (Case I)* $\gamma_2 = 5$, obtained with the full scheme (PBC, IC1, SA). The resulting relative L^2 error is 8.29×10^{-4} and the relative L^1 error is 5.34×10^{-4}. Bottom: Best solution of the *Allen–Cahn Equation (Case II)* $\gamma_2 = 4$, obtained with the full scheme (PBC, IC1, SA). The resulting relative L^2 error is 3.53×10^{-4} and the relative L^1 error is 2.14×10^{-4}. Full (x, t) network predictions are in Fig. 1.

contrast to the Allen–Cahn equation discussed in Section 4.2, the choice of initial condition plays a more significant role in the training process for this more complex system. This observation underscores the importance of carefully selecting both the boundary conditions and initial conditions when training neural networks for chaotic or higher-order PDE systems, where these factors contribute notably to the accuracy and stability of the solution.

4.4 Cahn–Hilliard equation

The Cahn–Hilliard equation, a variant of the Allen–Cahn equation, also models phase separation. This process describes how the components of a binary fluid spontaneously separate into distinct domains, each composed of a single component [6, 10, 27, 31, 39]. The equation is expressed as:

$$\begin{aligned} u_t &= \epsilon_1(-u_{xx} - \epsilon_2 u_{xxxx} + (u^3)_{xx}), \quad (t,x) \in (0,T] \times (-L,L), \\ u(0,x) &= u_0(x), \quad x \in [-L,L], \\ u(t,-L) &= u(t,L), \quad t \in [0,T], \end{aligned} \tag{10}$$

where $\epsilon_1 = 10^{-2}, \epsilon_2 = 10^{-4}, T = 1, L = 1$. We specify the initial condition as $u_0(x) = -\cos(2\pi x)$. This PDE involves higher-order derivatives, making it more challenging to solve compared to the Allen–Cahn equation.

In this experiment, the full scheme yields the best results. Additionally, methods using initial condition *IC2* outperform those with *IC1*. Except for the full schemes and the experiment with periodic neural networks for boundary conditions and initial condition *IC2*, none of the other approaches produce accurate solutions. The errors for all experiments are summarized in Table 3. The best results are illustrated in Figure 4.

Ablation Settings				Performance	
PBC	**IC1**	**IC2**	**SA**	**Rel. L^2 error**	**Rel. L^1 error**
✓	✓	✗	✓	4.04×10^{-2}	1.46×10^{-2}
✓	✗	✓	✓	5.37×10^{-4}	1.02×10^{-3}
✓	✓	✗	✗	1.68×10^{-2}	8.94×10^{-3}
✓	✗	✓	✗	1.30×10^{-2}	7.79×10^{-3}
✓	✗	✗	✗	9.84×10^{-3}	6.36×10^{-3}
✓	✗	✗	✓	2.50×10^{-3}	1.81×10^{-3}
✗	✓	✗	✗	2.13×10^{-1}	9.74×10^{-2}
✗	✓	✗	✓	2.00×10^{-1}	1.04×10^{-1}
✗	✗	✓	✗	1.54×10^{-2}	1.21×10^{-2}
✗	✗	✓	✓	9.60×10^{-3}	8.75×10^{-3}
✗	✗	✗	✓	2.11×10^{-1}	1.06×10^{-1}
✗	✗	✗	✗	3.01×10^{-1}	1.97×10^{-1}

Table 2: *Kuramoto–Sivashinsky:* Relative L^2 and L^1 errors for an ablation study illustrating the impact of disabling individual components of the proposed technique and training pipeline.

4.5 Gray–Scott equation

Reaction and diffusion of chemical species can produce a variety of patterns [14, 40], reminiscent of those often seen in nature. The Gray–Scott type system is one of classical mathematical models for chemical reactions [18, 19, 33]. The general irreversible Gray–Scott equations describe such reactions:

$$\begin{aligned} U + 2V &\longrightarrow 3V, \\ V &\longrightarrow P. \end{aligned} \tag{11}$$

This system is defined by two equations that describe the dynamics of two reacting substances:

$$\begin{aligned} u_t &= \epsilon_1 u_{xx} + b(1-u) - uv^2, \\ v_t &= \epsilon_2 v_{xx} - (b+k)v + uv^2, \quad (t,x) \in (0,T] \times (-L,L), \\ u(0,x) &= u_0(x), \quad v(0,x) = v_0(x), \quad x \in [-L,L], \\ u(t,-L) &= u(t,L), \quad v(t,-L) = v(t,L), \quad t \in [0,T], \end{aligned} \tag{12}$$

where $T = 20, L = 50$, $\epsilon_1 = 1, \epsilon_2 = 0.01$ are diffusion rates, $b = 0.02$ is the "feeding rate" that adds U, $k = 0.0562$ is the "killing rate" that removes V. We set our initial conditions as:

$$u_0(x) = 1 - \frac{\sin(\pi(x-50)/100)^4}{2}, \quad v_0(x) = \frac{\sin(\pi(x-50)/100)^4}{4}.$$

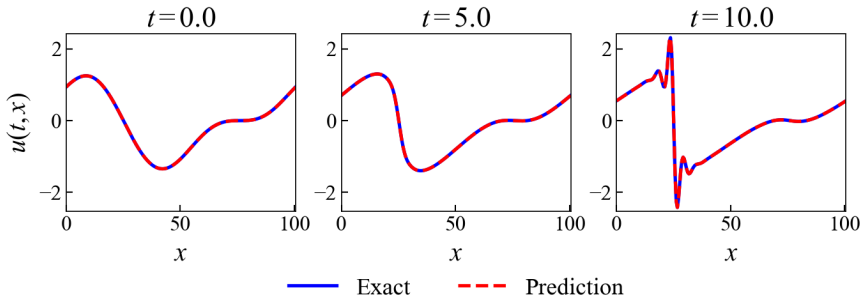

Figure 3: Best solution: *Kuramoto–Sivashinsky*, obtained with the configuration (PBC, IC2, SA). The resulting relative L^2 error is 5.37×10^{-4} and the relative L^1 error is 1.02×10^{-3}. Full (x, t) network predictions are in Fig. 1.

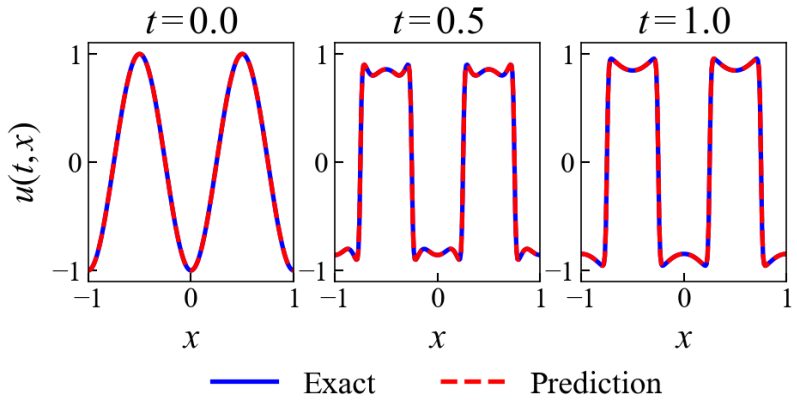

Figure 4: Best solution of the *Cahn–Hilliard*, obtained with the configuration (PBC, IC2, SA). The resulting relative L^2 error is 9.75×10^{-4} and the relative L^1 error is 5.65×10^{-4}. Full (x, t) network predictions are in Fig. 1.

In this ablation study, we not only evaluate the performance of self-adaptive neural networks but also test our scheme with mini-batching on the Gray–Scott equations, which model the densities of two interacting species. For mini-batching, we use 5 batches, each containing 3,280 samples. As shown in Table 4, the results confirm that each proposed component improves the overall model performance. Omitting any one of these components leads to higher error rates. Notably, the errors for the $u-$equation are typically smaller than those for the $v-$equation, which aligns with our expectations. This is because the $v-$equation is sharper and more challenging to train compared to the $u-$equation. Additionally, experiments using initial condition *IC2* consistently outperform those using *IC1*. Specifically, the full scheme with *IC2* achieves the best results, with a relative L^2 error of 2.05×10^{-4} for $u-$equation and 8.46×10^{-4} for $v-$equation. These results suggest that *IC2* is more effective for larger time-scale PDEs, as *IC1* causes the initial condition to decay too quickly, reducing its influence during training. The predicted solutions from our best model are visualized in Fig. 5.

Ablation Settings				Performance	
PBC	**IC1**	**IC2**	**SA**	**Rel. L^2 error**	**Rel. L^1 error**
✓	✓	✗	✓	1.06×10^{-3}	5.61×10^{-4}
✓	✗	✓	✓	9.75×10^{-4}	5.65×10^{-4}
✓	✓	✗	✗	2.50×10^{-1}	5.87×10^{-2}
✓	✗	✓	✗	3.87×10^{-3}	1.91×10^{-3}
✓	✗	✗	✗	9.99×10^{-1}	9.99×10^{-1}
✓	✗	✗	✓	9.99×10^{-1}	9.99×10^{-1}
✗	✓	✗	✗	3.48×10^{-1}	2.94×10^{-1}
✗	✓	✗	✓	2.37×10^{-1}	1.09×10^{-1}
✗	✗	✓	✗	4.07×10^{-1}	3.45×10^{-1}
✗	✗	✓	✓	1.55×10^{-2}	8.20×10^{-3}
✗	✗	✗	✓	9.99×10^{-1}	9.99×10^{-1}
✗	✗	✗	✗	9.99×10^{-1}	9.98×10^{-1}

Table 3: *Cahn–Hilliard Equation:* Relative L^2 and L^1 errors for an ablation study illustrating the impact of disabling individual components of the proposed technique and training pipeline.

4.6 Nonlinear Schrödinger equation

In theoretical physics, nonlinear Schrödinger equation is a nonlinear PDE, applicable to classical and quantum mechanics [17, 25, 26]. The dimensionless equation of the classical field is

$$\begin{aligned} u_t &= iu_{xx} + i|u|^2 u, \quad (t,x) \in [0,2] \times [-\pi,\pi], \\ u(0,x) &= u_0(x), \quad x \in [-\pi,\pi], \\ u(t,-\pi) &= u(t,\pi), \quad t \in [0,2], \end{aligned} \tag{13}$$

where we let

$$u_0(x) = \frac{2}{2-\sqrt{2}\cos(x)} - 1.$$

Using the same parameter settings and training steps with prior experiments, we obtain the following results:

Table 5 presents the results of all experiments. The full scheme achieves the best performance, with a relative L^2 error of real part of 1.80×10^{-4} and the relative L^2 error of imaginary part of 2.58×10^{-4}. There is no significant difference in the errors between methods using *IC1* and those using *IC2*. Moreover, without the use of self-adaptive neural networks or mini-batching, accurately obtaining the solution becomes challenging, highlighting the importance of both methods in ensuring more stable training.

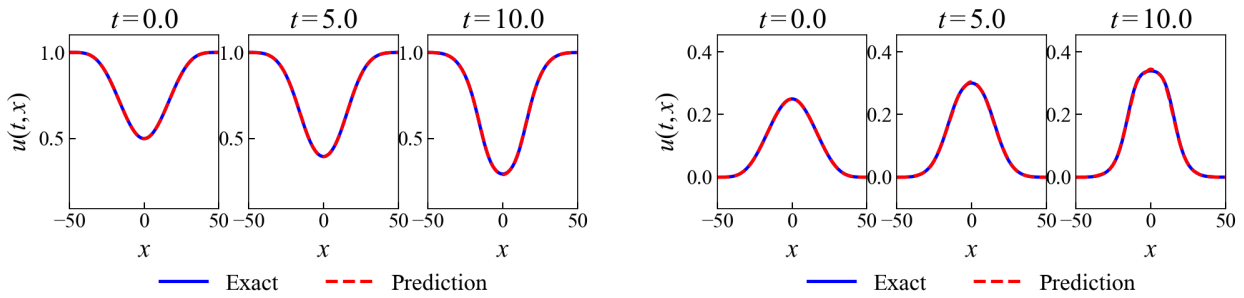

Figure 5: Best solution of the *Gray–Scott*, obtained with the configuration (PBC, IC2, SA, mini-batch). The resulting relative L^2 errors of u-species and v-species are 2.05×10^{-4} and 8.46×10^{-4} respectively. Full (x, t) network predictions are in Fig. 1.

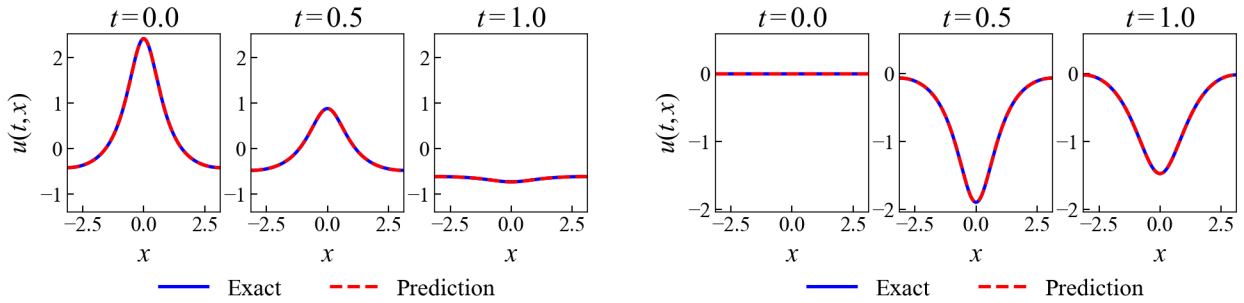

Figure 6: Best solution of the *Nonlinear Schrödinger Equation*, obtained with the configuration (PBC, IC1, SA, mini-batching). The resulting relative L^2 errors of real-part and imaginary-part solutions are 1.80×10^{-4} and 2.58×10^{-4} respectively. Full (x, t) network predictions are in Fig. 1.

4.7 Linear advection

The linear advection equation models the transport of a quantity by a constant velocity field and serves as a fundamental benchmark for testing numerical methods due to its simplicity and rich structure [32, 49]. It describes the propagation of a scalar profile without deformation and is given by:

$$\begin{aligned} u_t + c u_x &= 0, && (t, x) \in (0, T] \times (0, 2\pi), \\ u(0, x) &= \sin(x), && x \in [0, 2\pi], \\ u(t, 0) &= u(t, 2\pi), && t \in [0, T]. \end{aligned} \tag{14}$$

Unlike dissipative PDEs such as the Allen–Cahn or Cahn–Hilliard equations, the linear advection equation conserves the shape of the initial profile as it translates. This problem becomes "stiffer" as the advection velocity increases.

Theorem 4.1 (Transformation for the linear advection equation)**.** *Consider the linear advection problem* (14)*. Let*

$$\tilde{u}(t, x) = \psi(t, x) + \phi(t, x) u_{nn}(t, x)$$

be the transformed neural approximation defined in (4)*, where* ψ *and* ϕ *satisfy the initial-condition requirements in* (5)*. Assume further that* ψ*,* ϕ*, and* u_{nn} *are periodic in* x *with period* 2π*. Then* $\tilde{u}$ *satisfies* (14) *if and only if* u_{nn} *satisfies*

$$\phi\big(\partial_t u_{nn} + c\,\partial_x u_{nn}\big) + \big(\phi_t + c\,\phi_x\big) u_{nn} = -\big(\psi_t + c\,\psi_x\big).$$

Ablation Settings					Performance	
PBC	**IC1**	**IC2**	**SA**	**mini-batch**	**Rel. L^2 error: u**	**Rel. L^2 error: v**
✓	✓	✗	✓	✓	8.98×10^{-4}	7.92×10^{-3}
✓	✗	✓	✓	✓	2.05×10^{-4}	8.46×10^{-4}
✓	✓	✗	✗	✓	4.93×10^{-3}	3.03×10^{-2}
✓	✗	✓	✗	✓	2.86×10^{-3}	8.20×10^{-3}
✓	✓	✗	✓	✗	9.01×10^{-3}	4.27×10^{-3}
✓	✗	✓	✓	✗	7.59×10^{-3}	3.98×10^{-3}
✓	✓	✗	✗	✗	5.11×10^{-3}	2.58×10^{-2}
✓	✗	✓	✗	✗	1.23×10^{-3}	9.74×10^{-3}
✓	✗	✗	✗	✗	8.09×10^{-1}	9.99×10^{-1}
✗	✓	✗	✗	✗	2.06×10^{-1}	7.43×10^{-1}
✗	✗	✓	✗	✗	5.37×10^{-1}	2.58×10^{-1}
✗	✗	✗	✓	✗	1.15×10^{-1}	2.62×10^{-1}
✗	✗	✗	✗	✓	9.53×10^{-2}	8.65×10^{-2}
✗	✗	✗	✗	✗	9.21×10^{-2}	2.98×10^{-1}

Table 4: *Gray–Scott Equation:* Relative L^2 errors of u and v equations for an ablation study illustrating the impact of disabling individual components of the proposed technique and training pipeline.

This transformed problem is well-posed and stable for PINN training provided the forcing $-(\psi_t + c\psi_x) \in C^1([0,T] \times \Omega)$ *and is uniformly bounded or decaying in time.*

Proof. By the transformation (4),

$$\begin{aligned} \partial_t \tilde{u} + c\,\partial_x \tilde{u} &= \partial_t(\psi + \phi u_{nn}) + c\,\partial_x(\psi + \phi u_{nn}) \\ &= \psi_t + c\,\psi_x + \phi\big(\partial_t u_{nn} + c\,\partial_x u_{nn}\big) + \big(\phi_t + c\,\phi_x\big)u_{nn}. \end{aligned}$$

Therefore,

$$\partial_t \tilde{u} + c\,\partial_x \tilde{u} = 0$$

if and only if

$$\phi\big(\partial_t u_{nn} + c\,\partial_x u_{nn}\big) + \big(\phi_t + c\,\phi_x\big)u_{nn} = -\big(\psi_t + c\,\psi_x\big).$$

Moreover, the assumptions in (5) imply

$$\tilde{u}(0,x) = \psi(0,x) + \phi(0,x)u_{nn}(0,x) = u_0(x),$$

and the periodicity of ψ, ϕ, and u_{nn} implies

$$\tilde{u}(t,0) = \tilde{u}(t,2\pi).$$

Hence $\tilde{u}$ satisfies the complete initial-boundary value problem (14) if and only if u_{nn} satisfies the transformed equation above. □

For our implementation, we take

$$\psi(t,x) = u_0(x)e^{-\alpha t}, \qquad \phi(t) = \frac{t}{T}, \qquad \alpha > 0.$$

Since $u_0(x) = \sin x$ is periodic, both ψ and $\tilde{u}$ are periodic in x under the periodic neural network construction for u_{nn}. Moreover,

$$\tilde{u}(0,x) = u_0(x), \qquad \phi_t(0,x) = \frac{1}{T} \neq 0,$$

so the transformation is non-degenerate at $t = 0$ in the sense of Theorem 3.1. Since $\phi_t = 1/T$ and $\phi_x = 0$, the transformed equation becomes

$$\frac{t}{T}\big(\partial_t u_{nn} + c\,\partial_x u_{nn}\big) + \frac{1}{T}u_{nn} = -\big(\psi_t + c\,\psi_x\big).$$

With $u_0(x) = \sin x$, the right-hand side is

$$-\big(\psi_t + c\,\psi_x\big) = \big(\alpha \sin x - c\cos x\big)e^{-\alpha t},$$

which is smooth and decays exponentially in time, guaranteeing both the stability and the regularity of the learned component $u_{nn}(t,x)$. Here, we compare the results produced by our method with those from a baseline PINN under the same hyperparameter settings. In our experiments, both models were able to converge at low velocities; however, we observed that higher advection velocities introduced stiffness. Our method exhibited greater robustness and was less affected by stiffness compared to the baseline PINN.

The implementation details are the same as in Section 4.1, except that we perform 10 independent replications of the experiment to account for the randomness in neural network weight initialization, rather than using the same initial weights. Table 6 shows the averaged relative L^1/L^2-error over 10 independent repetitions. As the advection velocity increases, the PDE becomes stiffer and harder for the baseline PINN to train. Our method, however, consistently produces better results, as clearly reflected in Table 6. Even at low velocities, our method demonstrates significantly better accuracy compared to the baseline PINN. As c increases, both the relative L^1 and L^2 errors grow, however, the errors associated with the PINN rise significantly faster than those of our method. The most striking example occurs at $c = 50$, where the baseline PINN fails to approximate the solution, while our method still produces stable and good results. This behavior is evident in Table 6 and Figure 7. Table 7 shows the ablation study of linear advection equation.

4.8 Comparison to other state-of-the-art PINNs

Figure 1 provides a summary of our ablation study over different benchmarks; it shows the comparison between the reference exact solutions and our best HC-PINN predictions, together with the corresponding absolute errors; the benchmarks include Allen–Cahn, Cahn–Hilliard, Kuramoto–Sivashinsky, Gray–Scott, and nonlinear Schrödinger. Complementing these visual comparisons,

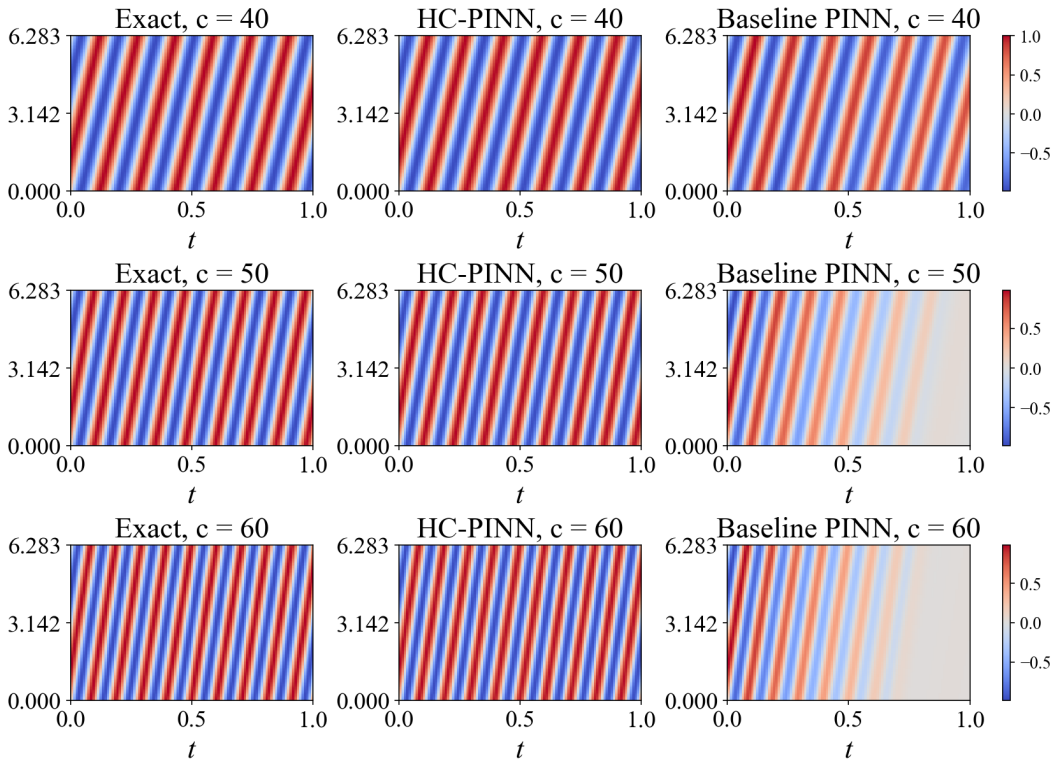

Figure 7: Periodic advection results after 50,000 iterations: exact solution, HC-PINN, and baseline PINN for $c = 40, 50, 60$ (cf. Table 6).

Table 8 summarizes the comparison study between HC-PINN and two other PINN variants in terms of relative L^2 errors for Allen–Cahn, Cahn–Hilliard and Kuramoto–Sivashinsky equations under a strictly controlled setting: all methods are trained using the same optimization budget: 50,000 Adam iterations followed by 5,000 L-BFGS-B iterations, same data and parameter settings to ensure fairness, unless otherwise specified. We emphasize that this controlled setting is intended to isolate the effect of the training scheme under an equal compute budget, and we note that the shorter budget may be unfavorable to the baseline methods; for transparency, the originally reported $300k$-iteration values of the baselines are included in Table 8 as a reference. Under this equal-budget protocol, HC-PINN attains substantially lower errors across all three benchmarks (while at full 300k budget Enhanced RBA matches or surpasses HC-PINN on AC Case I, our method remains substantially more accurate at the same compute budget). Overall, these extensive experiments highlight the effectiveness and adaptability of our PINN scheme in solving a wide variety of stiff time-dependent PDEs. The results provide not only strong empirical validation but also a reliable and reproducible benchmark for future developments in this area.

	Original ($300k$)		Equal budget ($50k$)		
	Causal	RBA	Causal (MLP)	RBA	HC-PINNs
AC (Case I)	1.43×10^{-3}	5.7×10^{-5}	6.95×10^{-2}	2.62×10^{-3}	8.29×10^{-4}
AC (Case II)	–	–	1.78×10^{-2}	1.08×10^{-3}	3.53×10^{-4}
CH	–	–	3.49×10^{-1}	9.83×10^{-2}	9.75×10^{-4}
KS	2.26×10^{-2}	–	2.72×10^{-1}	3.64×10^{-2}	5.37×10^{-4}

Table 8: Relative L^2 errors for the Allen–Cahn (AC), Cahn–Hilliard (CH) and Kuramoto–Sivashinsky (KS) equations. The right block reports results under a strictly controlled equal-budget protocol in which all methods are trained for $50k$ iterations followed by 5,000 L-BFGS-B iterations with the same data and parameter settings. The left block reports the originally published $300k$-iteration values of Causal training [51] and Enhanced RBA [2], included as a transparent reference. Entries marked $-$ indicate that the corresponding benchmark or case was not reported in the original paper at $300k$ iterations. All HC-PINN entries are relative L^2 errors of the best-performing configuration: AC (Case I) and AC (Case II) use (PBC, IC1, SA), CH uses (PBC, IC2, SA), and KS uses (PBC, IC2, SA), consistent with the corresponding ablation tables and best-solution figures.

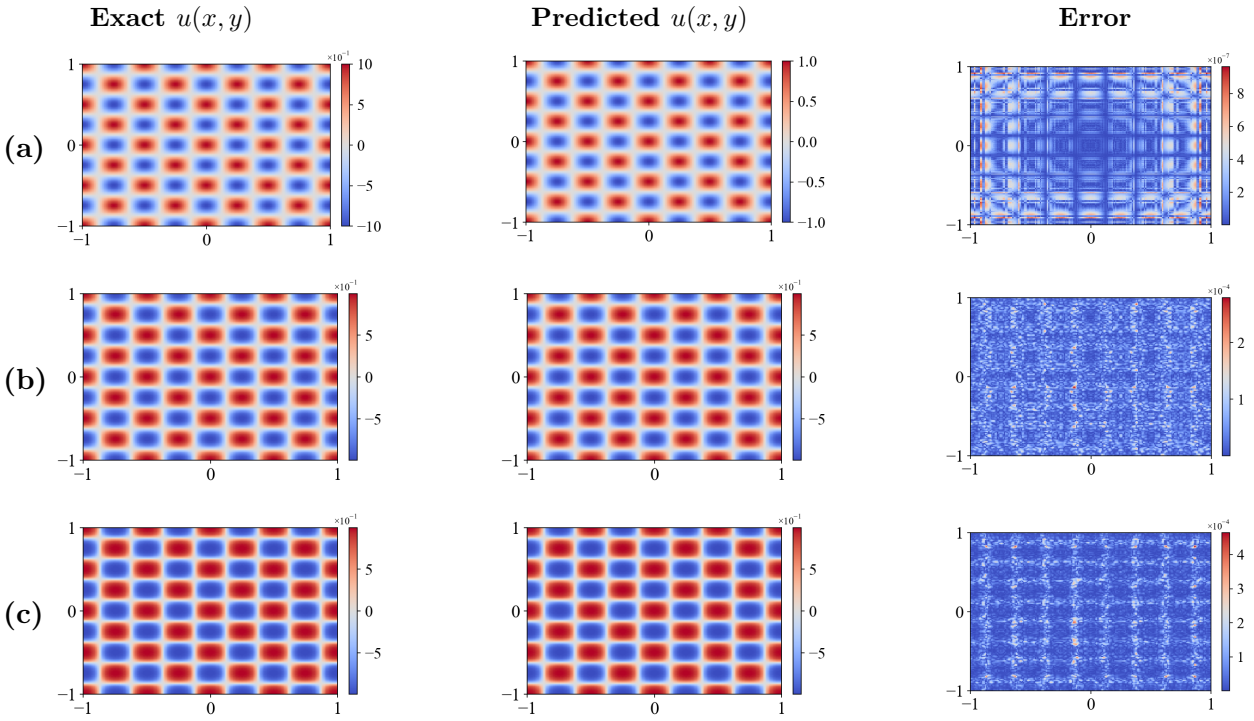

Figure 8: Reference, HC-PINN predicted solution, and absolute error of the $2D$ Allen Cahn equation at different time snapshots (a) $t = 0$, (b) $t = 0.5$, (c) $t = 1$.

4.9 Two-dimensional Allen–Cahn equation and sensitivity test

The results of solving the $2D$ Allen–Cahn equation in Figure 8 were obtained after training the network for $50k$ iterations using the initial condition $u(0, x, y) = \sin(4\pi x)\cos(4\pi y)$, with $\gamma_1 = 0.0001$ and $\gamma_2 = 1$ in Eq. 8, and periodic boundary conditions. The HC-PINN accurately captures the solution evolution, yielding exceptionally small relative L^2 errors: 8.04×10^{-5} at $t = 0.5$ and 1.33×10^{-4} at $t = 1$. This demonstrates the high accuracy and stability of our method for 2D Allen–Cahn dynamics. At last, we perform a sensitivity study on the number of Fourier modes m for the Allen–Cahn benchmarks to assess how this key hyperparameter influences the accuracy of HC-PINNs. The results demonstrate that accuracy is highly dependent on spectral resolution: while small m yields stable and low errors, performance deteriorates significantly once m exceeds a critical threshold.

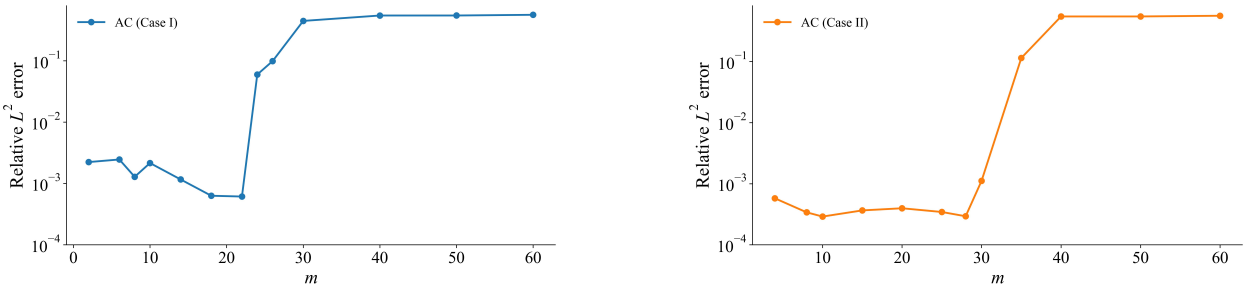

Figure 9: Sensitivity of HC-PINNs to the number of Fourier modes m in the Allen–Cahn equations. The plot shows the relative L^2 error as a function of m.

4.10 Robustness to non-periodic boundary conditions

All examples in Sections 4.2–4.7 use periodic boundary conditions and rely on the Fourier-feature architecture to enforce periodicity in u_{nn}. To assess whether the stabilizing effect documented in those sections is intrinsic to the hard-initial-condition mechanism rather than to the periodic architecture, we test HC-PINN on the same Allen–Cahn problem of Case I (Section 4.2) but with non-homogeneous Dirichlet boundary conditions:

$$\begin{aligned} u_t - \gamma_1 u_{xx} + \gamma_2(u^3 - u) = 0, \quad (t, x) \in (0, 1] \times (-1, 1), \\ u(0, x) = x^2 \cos(\pi x), \quad x \in [-1, 1], \\ u(t, -1) = u(t, 1) = -1, \quad t \in [0, 1], \end{aligned} \tag{15}$$

with $\gamma_1 = 10^{-3}$ and $\gamma_2 = 5$ as in Case I. The boundary value $g = -1$ is dictated by the compatibility condition $u_0(\pm 1) = -1$, so the same initial profile from the periodic Case I is reused without modification.

The general hard-constraint construction (4) admits the boundary-aware lift

$$\psi(t, x) = u_0(x)\, e^{-t} + g\left(1 - e^{-t}\right), \qquad \phi(t, x) = t\,(1 - x^2),$$

which satisfies $\psi(0, x) = u_0(x)$, $\psi(t, \pm 1) = g$ for all t, $\phi(0, x) = 0$, and $\phi(t, \pm 1) = 0$. Hence $\tilde{u}(0, x) = u_0(x)$ and $\tilde{u}(t, \pm 1) = g$ are enforced exactly by construction, irrespective of u_{nn}. In particular, neither an initial-condition loss nor a boundary-condition loss is needed, and the training reduces to a single PDE-residual objective.

Crucially, in this experiment $u_{nn}(t,x)$ is a standard fully connected MLP taking (t,x) directly as input, without any Fourier-feature layer. The architecture, optimizer schedule, and collocation budget are identical to the periodic experiments of Section 4.1.

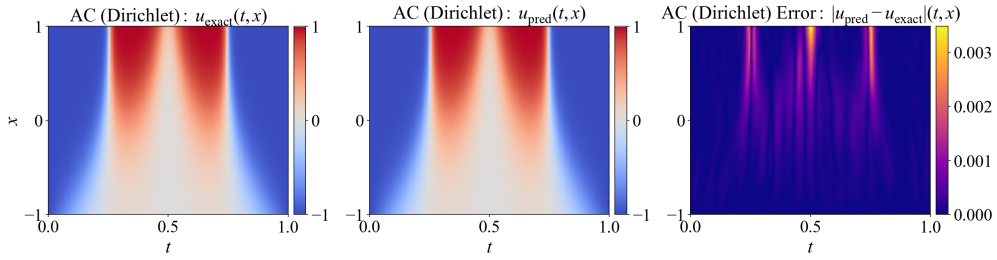

Figure 10: Allen–Cahn with non-homogeneous Dirichlet BC ($g=-1$), $\gamma_1=10^{-3}$, $\gamma_2=5$. Left: reference solution by Crank–Nicolson IMEX FD. Middle: HC-PINN prediction with plain MLP (no Fourier features) and hard IC/BC enforcement. Right: pointwise absolute error. Relative L^2 error 5.05×10^{-4}, relative L^1 error 2.82×10^{-4}.

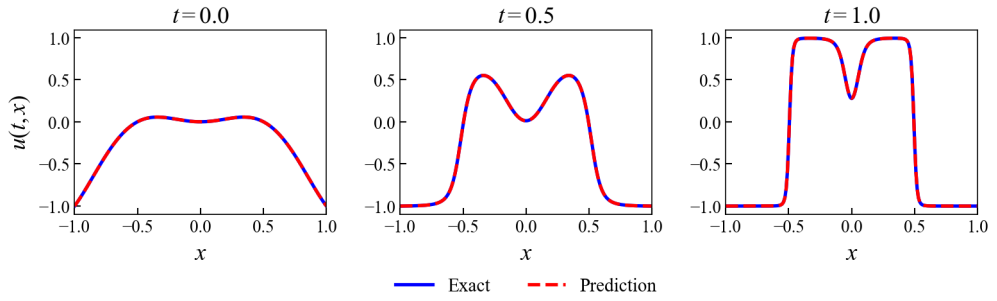

Figure 11: Allen–Cahn (Dirichlet): cross-sectional comparisons of the HC-PINN prediction against the reference solution at $t=0$, $t=0.5$, and $t=1.0$.

HC-PINN attains a relative L^2 error of 5.05×10^{-4} and a relative L^1 error of 2.82×10^{-4}, slightly lower than the periodic Case I error of 8.29×10^{-4} reported in Table 1. Figure 10 shows the reference solution, the HC-PINN prediction, and the pointwise absolute error: the network captures the sharp interface dynamics and the boundary layers induced by the Dirichlet data without any visible artifacts at the boundary.

This experiment confirms that the stabilizing role of exact initial-condition enforcement is independent of the periodic Fourier-feature architecture: the same hard-constraint mechanism, boundary-compatible choice of ψ and a multiplier ϕ that vanishes on $\partial\Omega$, transfers cleanly to the Dirichlet setting. A systematic study across Neumann, Robin, and mixed boundary types on more complex geometries is left to future work.

5 Conclusion

We provide a comprehensive study of hard constraints and self-adaptive loss weighting for stabilizing the training of PINNs in solving stiff time-dependent PDEs. Our study includes an ablation analysis across seven representative PDEs, a comparison with state-of-the-art methods, a theoretical investigation, an application to the $2D$ Allen–Cahn equation, and a sensitivity analysis with respect to m (the number of Fourier modes).

The ablation results demonstrate that the hard-constrained transformation plays a dominant role in alleviating the training challenges associated with stiff time-dependent PDEs. In particular, it consistently improves performance across a diverse set of problems, including scalar-valued and vector-valued (including complex-valued) systems. This transformation enhances both training stability and solution accuracy, especially in regimes where conventional PINNs often fail. From a theoretical perspective, our NTK analysis provides insight into this improvement by showing that the hard-constrained formulation mitigates spectral bias and simplifies the training dynamics.

The comparison study further confirms that HC-PINN outperforms existing state-of-the-art approaches on standard benchmarks, highlighting its effectiveness and robustness. Overall, our findings suggest that enforcing initial conditions through hard constraints is not merely beneficial, but essential for reliable PINN training in stiff regimes.

The reported experiments are predominantly periodic, with one non-periodic Dirichlet example (Section 4.10) indicating that the hard-IC mechanism is the dominant stabilizing factor irrespective of boundary type. A systematic study across Neumann, Robin, and mixed boundaries on complex geometries is left to future work. Future work includes extending the hard-constrained framework to more complex two- and three-dimensional PDEs, where additional challenges arise in both architecture design and computational cost. Another important direction is the systematic design of optimal transformations ψ and ϕ, which govern the embedding of initial and boundary conditions into the network structure, with the goal of further improving performance and generalizability.

Author contributions

Baoli Hao: Conceptualization, Methodology, Software, Investigation, Formal analysis, Visualization, Writing - original draft; **Ming Zhong:** Conceptualization, Methodology, Supervision, Writing - review & editing; **Ulisses Braga-Neto:** Conceptualization, Methodology; **Chun Liu:** Supervision, Writing - review & editing; **Lifan Wang:** Conceptualization, Writing - review & editing. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

The authors BH and MZ would like to thank Prof. Yannis Kevrekidis (JHU) for the discussion on the well-conditioning of PDE solution maps, and Prof. George Karniadakis (Brown) for his suggestions on training enhancement methods for PINNs. MZ is supported by NSF-AoF grant #2225507 and Ralph E. Powe Junior Faculty Enhancement Award from ORAU for FY24.

Use of Generative-AI tools declaration

The authors used AI-based language tools to improve clarity and grammar. No AI tools were used for scientific content generation.

A Proofs of the Theorems

We present the proofs of the two main theoretical results in Section 3. The elementary derivation associated with Theorem 4.1 is provided directly in Section 4.7.

Proof of Theorem 3.1 (Consistency and Non-Degeneracy of the Transformation)

Proof. By the definition of the hard-constraint transformation,

$$u(t,\mathbf{x}) = \psi(t,\mathbf{x}) + \phi(t,\mathbf{x})u_{nn}(t,\mathbf{x}).$$

Using

$$\psi(0,\mathbf{x}) = u_0(\mathbf{x}), \qquad \phi(0,\mathbf{x}) = 0, \qquad \mathbf{x} \in \bar{\Omega},$$

we immediately obtain

$$u(0,\mathbf{x}) = \psi(0,\mathbf{x}) + \phi(0,\mathbf{x})u_{nn}(0,\mathbf{x}) = u_0(\mathbf{x}).$$

Thus, the prescribed initial condition is satisfied exactly.

Next, differentiating the transformation with respect to time gives

$$\partial_t u = \psi_t + \phi\partial_t u_{nn} + \phi_t u_{nn}.$$

Therefore,

$$\partial_t u + \mathcal{P}[u] - f = \psi_t + \phi\partial_t u_{nn} + \phi_t u_{nn} + \mathcal{P}[\psi + \phi u_{nn}] - f.$$

Hence,

$$\partial_t u + \mathcal{P}[u] = f$$

if and only if

$$\phi\partial_t u_{nn} + \mathcal{P}[\psi + \phi u_{nn}] + \phi_t u_{nn} = f - \psi_t.$$

This proves the equivalence between the original PDE and the transformed PDE.

Finally, evaluating the time derivative of the transformation at $t = 0$ and using $\phi(0,\mathbf{x}) = 0$, we obtain

$$u_t(0,\mathbf{x}) = \psi_t(0,\mathbf{x}) + \phi_t(0,\mathbf{x})u_{nn}(0,\mathbf{x}).$$

Since

$$\phi_t(0,\mathbf{x}) \neq 0, \qquad \mathbf{x} \in \bar{\Omega},$$

the initial value of the transformed variable is uniquely determined by

$$u_{nn}(0,\mathbf{x}) = \frac{u_t(0,\mathbf{x}) - \psi_t(0,\mathbf{x})}{\phi_t(0,\mathbf{x})}.$$

Therefore, the transformation is non-degenerate at the initial time in the sense stated in the theorem. □

Proof of Theorem 3.3 (Hessian Characterization and Bounds)

Proof. Since

$$u(\mathbf{y};\boldsymbol{\theta}) = \mathbf{h}^\top(\mathbf{y})\boldsymbol{\theta},$$

the loss of the soft-constrained PINN can be written as

$$\begin{aligned}\mathrm{Loss}_s(u(\boldsymbol{\theta})) = \frac{1}{2N}\sum_{i=1}^{N} |\big(\mathcal{L}_s[u(\cdot;\boldsymbol{\theta})] - f\big)(\mathbf{z}_i)|^2 \\ + \frac{1}{2M}\sum_{j=1}^{M} |u(\boldsymbol{\zeta}_j;\boldsymbol{\theta}) - u_0(\mathbf{x}_j^{IC})|^2.\end{aligned}$$

Define

$$\mathbf{A} = \begin{bmatrix} \big(\mathcal{L}_s[\mathbf{h}^\top]\big)(\mathbf{z}_1) \\ \vdots \\ \big(\mathcal{L}_s[\mathbf{h}^\top]\big)(\mathbf{z}_N) \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} \mathbf{h}^\top(\boldsymbol{\zeta}_1) \\ \vdots \\ \mathbf{h}^\top(\boldsymbol{\zeta}_M) \end{bmatrix},$$

and

$$\mathbf{f} = \begin{bmatrix} f(\mathbf{z}_1) \\ \vdots \\ f(\mathbf{z}_N) \end{bmatrix}, \qquad \mathbf{u}_0 = \begin{bmatrix} u_0(\mathbf{x}_1^{IC}) \\ \vdots \\ u_0(\mathbf{x}_M^{IC}) \end{bmatrix}.$$

Then

$$\mathrm{Loss}_s(u(\boldsymbol{\theta})) = \frac{1}{2N}|\mathbf{A}\boldsymbol{\theta} - \mathbf{f}|^2_{\ell^2(\mathbb{R}^N)} + \frac{1}{2M}|\mathbf{C}\boldsymbol{\theta} - \mathbf{u}_0|^2_{\ell^2(\mathbb{R}^M)}.$$

For the hard-constrained formulation, recall that

$$\widetilde{u} = \psi + \phi u.$$

Since

$$\mathcal{L}_h[u] = \phi\partial_t u + \mathcal{P}[\psi + \phi u] + \phi_t u,$$

we have

$$\mathcal{L}_h[u] - f + \psi_t = \mathcal{L}_s[\psi + \phi u] - f.$$

Therefore, the hard-constrained loss can equivalently be written as

$$\mathrm{Loss}_h(u(\boldsymbol{\theta})) = \frac{1}{2N}\sum_{i=1}^{N} |\big(\mathcal{L}_s[\psi(\cdot) + \phi(\cdot)u(\cdot;\boldsymbol{\theta})] - f\big)(\mathbf{z}_i)|^2.$$

Since $\mathcal{P}$ is linear and $\phi(t,\mathbf{x}) = \phi(t)$, the operator $\mathcal{P}$ acts only on the spatial variables and hence

$$\mathcal{P}[\phi u] = \phi\mathcal{P}[u].$$

Consequently,

$$\begin{aligned} \mathcal{L}_s[\psi + \phi u] &= \partial_t(\psi + \phi u) + \mathcal{P}[\psi + \phi u] \\ &= \psi_t + \phi_t u + \phi u_t + \mathcal{P}[\psi] + \phi\mathcal{P}[u] \\ &= \phi\big(u_t + \mathcal{P}[u]\big) + \phi_t u + \psi_t + \mathcal{P}[\psi] \\ &= \phi\mathcal{L}_s[u] + \phi_t u + \mathcal{L}_s[\psi]. \end{aligned}$$

Define

$$\mathbf{B} = \begin{bmatrix} \mathbf{h}^\top(\mathbf{z}_1) \\ \vdots \\ \mathbf{h}^\top(\mathbf{z}_N) \end{bmatrix}, \qquad \tilde{\mathbf{f}} = \begin{bmatrix} f(\mathbf{z}_1) - \mathcal{L}_s[\psi](\mathbf{z}_1) \\ \vdots \\ f(\mathbf{z}_N) - \mathcal{L}_s[\psi](\mathbf{z}_N) \end{bmatrix},$$

and

$$\boldsymbol{\Lambda}_\phi = \mathrm{diag}\big(\phi(\mathbf{z}_1), \ldots, \phi(\mathbf{z}_N)\big), \qquad \boldsymbol{\Lambda}_{\phi_t} = \mathrm{diag}\big(\phi_t(\mathbf{z}_1), \ldots, \phi_t(\mathbf{z}_N)\big).$$

It follows that

$$\mathrm{Loss}_h(u(\boldsymbol{\theta})) = \frac{1}{2N}|\left(\boldsymbol{\Lambda}_\phi\mathbf{A} + \boldsymbol{\Lambda}_{\phi_t}\mathbf{B}\right)\boldsymbol{\theta} - \tilde{\mathbf{f}}|^2_{\ell^2(\mathbb{R}^N)}.$$

The corresponding gradients are

$$\nabla_{\boldsymbol{\theta}}\mathrm{Loss}_s(u(\boldsymbol{\theta})) = \frac{1}{N}\left(\mathbf{A}^\top\mathbf{A}\boldsymbol{\theta} - \mathbf{A}^\top\mathbf{f}\right) + \frac{1}{M}\left(\mathbf{C}^\top\mathbf{C}\boldsymbol{\theta} - \mathbf{C}^\top\mathbf{u}_0\right),$$
$$\nabla_{\boldsymbol{\theta}}\mathrm{Loss}_h(u(\boldsymbol{\theta})) = \frac{1}{N}\Big[\left(\boldsymbol{\Lambda}_\phi\mathbf{A} + \boldsymbol{\Lambda}_{\phi_t}\mathbf{B}\right)^\top\left(\boldsymbol{\Lambda}_\phi\mathbf{A} + \boldsymbol{\Lambda}_{\phi_t}\mathbf{B}\right)\boldsymbol{\theta} - \left(\boldsymbol{\Lambda}_\phi\mathbf{A} + \boldsymbol{\Lambda}_{\phi_t}\mathbf{B}\right)^\top\tilde{\mathbf{f}}\Big].$$

Differentiating once more with respect to $\boldsymbol{\theta}$, we obtain

$$H_s(\boldsymbol{\theta}) = \frac{1}{N}\mathbf{A}^\top\mathbf{A} + \frac{1}{M}\mathbf{C}^\top\mathbf{C},$$

and

$$\begin{aligned} H_h(\boldsymbol{\theta}) &= \frac{1}{N}\left(\boldsymbol{\Lambda}_\phi\mathbf{A} + \boldsymbol{\Lambda}_{\phi_t}\mathbf{B}\right)^\top\left(\boldsymbol{\Lambda}_\phi\mathbf{A} + \boldsymbol{\Lambda}_{\phi_t}\mathbf{B}\right) \\ &= \frac{1}{N}\Big(\mathbf{A}^\top\boldsymbol{\Lambda}_\phi^2\mathbf{A} + \mathbf{A}^\top\boldsymbol{\Lambda}_\phi\boldsymbol{\Lambda}_{\phi_t}\mathbf{B} + \mathbf{B}^\top\boldsymbol{\Lambda}_{\phi_t}\boldsymbol{\Lambda}_\phi\mathbf{A} + \mathbf{B}^\top\boldsymbol{\Lambda}_{\phi_t}^2\mathbf{B}\Big). \end{aligned}$$

We next derive the Hessian bounds. Here, $|\mathcal{L}_s|_2$ denotes an induced discrete operator bound on the sampled linearized trial space such that

$$|\mathbf{A}|_2 \le |\mathcal{L}_s|_2|\mathbf{B}|_2.$$

Therefore,

$$|\mathbf{A}^\top\mathbf{A}|_2 = |\mathbf{A}|_2^2 \le |\mathcal{L}_s|_2^2|\mathbf{B}|_2^2.$$

It follows immediately that

$$\begin{aligned} |H_s|_2 &\le \frac{1}{N}|\mathbf{A}^\top\mathbf{A}|_2 + \frac{1}{M}|\mathbf{C}^\top\mathbf{C}|_2 \\ &\le \frac{|\mathcal{L}_s|_2^2}{N}|\mathbf{B}|_2^2 + \frac{1}{M}|\mathbf{C}|_2^2. \end{aligned}$$

Define

$$C_1 = \max_{\mathbf{z}}\phi^2(\mathbf{z}), \qquad C_2 = \max_{\mathbf{z}}\left|\phi(\mathbf{z})\phi_t(\mathbf{z})\right|, \qquad C_3 = \max_{\mathbf{z}}\phi_t^2(\mathbf{z}).$$

Then

$$|\mathbf{A}^\top\boldsymbol{\Lambda}_\phi^2\mathbf{A}|_2 \le C_1|\mathbf{A}|_2^2 \le C_1|\mathcal{L}_s|_2^2|\mathbf{B}|_2^2,$$
$$|\mathbf{A}^\top\boldsymbol{\Lambda}_\phi\boldsymbol{\Lambda}_{\phi_t}\mathbf{B}|_2 \le C_2|\mathbf{A}|_2|\mathbf{B}|_2 \le C_2|\mathcal{L}_s|_2|\mathbf{B}|_2^2,$$
$$|\mathbf{B}^\top\boldsymbol{\Lambda}_{\phi_t}\boldsymbol{\Lambda}_\phi\mathbf{A}|_2 \le C_2|\mathbf{B}|_2|\mathbf{A}|_2 \le C_2|\mathcal{L}_s|_2|\mathbf{B}|_2^2,$$
$$|\mathbf{B}^\top\boldsymbol{\Lambda}_{\phi_t}^2\mathbf{B}|_2 \le C_3|\mathbf{B}|_2^2.$$

Combining these estimates yields

$$|H_h|_2 \le \frac{1}{N}\left(C_1|\mathcal{L}_s|_2^2 + 2C_2|\mathcal{L}_s|_2 + C_3\right)|\mathbf{B}|_2^2.$$

This completes the proof. □

References

[1] [10.1016/0001-6160(79)90196-2] S. M. Allen and J. W. Cahn, *A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening*, *Acta Metallurgica*, **27** (1979), 1085-1095.

[2] (MR4698526) [10.1016/j.cma.2024.116805] S. J. Anagnostopoulos, J. D. Toscano, N. Stergiopoulos and G. E. Karniadakis, *Residual-based attention in physics-informed neural networks*, *Computer Methods in Applied Mechanics and Engineering*, **421** (2024), 116805.

[3] [10.1039/C7FD00037E] M. Z. Bazant, *Thermodynamic stability of driven open systems and control of phase separation by electro-autocatalysis*, *Faraday Discussions*, **199** (2017), 423-463.

[4] U. Braga-Neto, Characteristics-informed neural networks for forward and inverse hyperbolic problems, preprint, arXiv:2212.14012, 2022.

[5] (MR27195) [10.1016/S0065-2156(08)70100-5] J. M. Burgers, *A mathematical model illustrating the theory of turbulence*, *Advances in Applied Mechanics*, **1** (1948), 171-199.

[6] [10.1063/1.1744102] J. W. Cahn and J. E. Hilliard, *Free energy of a nonuniform system. I. Interfacial free energy*, *The Journal of Chemical Physics*, **28** (1958), 258-267.

[7] S. Cai, Z. Wang, L. Lu, T. A. Zaki and G. E. Karniadakis, DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks, *Journal of Computational Physics*, **436** (2021), 110296.

[8] [10.1016/j.csbj.2021.08.011] P. Carracedo-Reboredo, J. Liñares-Blanco, N. Rodríguez-Fernández, F. Cedrón, F. J. Novoa, A. Carballal, V. Maojo, A. Pazos and C. Fernandez-Lozano, *A review on machine learning approaches and trends in drug discovery*, *Computational and Structural Biotechnology Journal*, **19** (2021), 4538-4558.

[9] (MR4228919) [10.1007/s10910-021-01223-9] A. Cassani, A. Monteverde and M. Piumetti, *Belousov-Zhabotinsky type reactions: The non-linear behavior of chemical systems*, *Journal of Mathematical Chemistry*, **59** (2021), 792-826.

[10] K. S. Chan, B. Hao, K. F. Lam and B. Stinner, On a phase field model for binary mixtures of micropolar fluids with non-matched densities and moving contact lines, *Interfaces and Free Boundaries*, online first, 2026. DOI: 10.4171/IFB/570.

[11] X. Chen, D. J. Jeffery, M. Zhong, L. McClenny, U. Braga-Neto and L. Wang, Using physics informed neural networks for supernova radiative transfer simulation, preprint, arXiv:2211.05219, 2022.

[12] (MR4311012) [10.1016/j.jcp.2021.110668] Y. Chen, B. Hosseini, H. Owhadi and A. M. Stuart, *Solving and learning nonlinear PDEs with Gaussian processes*, *Journal of Computational Physics*, **447** (2021), 110668.

[13] (MR4602862) [10.1016/j.jcp.2023.112265] E. J. R. Coutinho, M. Dall'Aqua, L. McClenny, M. Zhong, U. Braga-Neto and E. Gildin, *Physics-informed neural networks with adaptive localized artificial viscosity*, *Journal of Computational Physics*, **489** (2023), 112265.

[14] [10.1016/0167-2789(91)90204-M] P. De. Kepper, V. Castets, E. Dulos and J. Boissonade, *Turing-type chemical patterns in the chlorite-iodide-malonic acid reaction*, *Physica D: Nonlinear Phenomena*, **49** (1991), 161-169.

[15] (MR4228390) [10.1016/j.jcp.2021.110242] S. Dong and N. Ni, *A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks*, *Journal of Computational Physics*, **435** (2021), 110242.

[16] [10.1038/s41586-022-05172-4] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R. Ruiz, J. Schrittwieser, G. Swirszcz, et al, *Discovering faster matrix multiplication algorithms with reinforcement learning*, *Nature*, **610** (2022), 47-53.

[17] (MR3308230) G. Fibich, *The Nonlinear Schrödinger Equation*, volume 192. Springer, 2015.

[18] [10.1016/0009-2509(83)80132-8] P. Gray and S. K. Scott, *Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Isolas and other forms of multistability*, *Chemical Engineering Science*, **38** (1983), 29-43.

[19] [10.1016/0009-2509(84)87017-7] P. Gray and S. K. Scott, *Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Oscillations and instabilities in the system* $\mathrm{A}+2\mathrm{B}\rightarrow 3\mathrm{B}$*;* $\mathrm{B}\rightarrow \mathrm{C}$, *Chemical Engineering Science*, **39** (1984), 1087-1097.

[20] K. Haitsiukevich and A. Ilin, Improved training of physics-informed neural networks with model ensembles, In *2023 International Joint Conference on Neural Networks (IJCNN)*, (2023), 1-8.

[21] [10.1039/c3ee24299d] B. Horstmann, T. Danner and W. G. Bessler, *Precipitation in aqueous lithium-oxygen batteries: A model-based analysis*, *Energy and Environmental Science*, **6** (2013), 1299-1314.

[22] [10.1146/annurev-cellbio-100913-013325] A. A. Hyman, C. A. Weber and F. Jülicher, *Liquid-liquid phase separation in biology*, *Annual Review of Cell and Developmental Biology*, **30** (2014), 39-58.

[23] (MR4195874) [10.1016/j.jcp.2020.109951] X. Jin, S. Cai, H. Li and G. E. Karniadakis, *NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations*, *Journal of Computational Physics*, **426** (2021), 109951.

[24] [10.1038/s41586-021-03819-2] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al, *Highly accurate protein structure prediction with AlphaFold*, *Nature*, **596** (2021), 583-589.

[25] (MR877998) T. Kato, On nonlinear Schrödinger equations, *Annales de l'I.H.P. Physique Théorique*, **46** (1987), 113-129.

[26] [10.1142/S0217979201007105] P. G. Kevrekidis, K. Ø. Rasmussen and A. R. Bishop, *The discrete nonlinear Schrödinger equation: A survey of recent results*, *International Journal of Modern Physics B*, **15** (2001), 2833-2900.

[27] (MR3568711) [10.1155/2016/9532608] J. Kim, S. Lee, Y. Choi, S.-M. Lee and D. Jeong, *Basic principles and practical applications of the Cahn-Hilliard equation*, *Mathematical Problems in Engineering*, **2016** (2016), 9532608.

[28] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby and M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, *Advances in Neural Information Processing Systems*, **34** (2021), 26548-26560.

[29] (MR1073332) [10.1016/0375-9601(90)90449-X] N. A. Kudryashov, *Exact solutions of the generalized Kuramoto-Sivashinsky equation*, *Physics Letters A*, **147** (1990), 287-291.

[30] [10.1109/72.712178] I. E. Lagaris, A. Likas and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, *IEEE Transactions on Neural Networks*, **9** (1998), 987-1000.

[31] [10.1016/j.commatsci.2013.08.027] D. Lee, J.-Y. Huh, D. Jeong, J. Shin, A. Yun and J. Kim, *Physical, mathematical, and numerical derivations of the Cahn-Hilliard equation*, *Computational Materials Science*, **81** (2014), 216-225.

[32] (MR1925043) R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, volume 31. Cambridge University Press, 2002.

[33] (MR4346701) [10.1016/j.jde.2021.11.032] J. Liang, N. Jiang, C. Liu, Y. Wang and T.-F. Zhang, *On a reversible Gray-Scott type system from energetic variational approach and its irreversible limit*, *Journal of Differential Equations*, **309** (2022), 427-454.

[34] [10.1115/1.4044400] D. Liu and Y. Wang, *Multi-fidelity physics-constrained neural network and its application in materials modeling*, *Journal of Mechanical Design*, **141** (2019), 121403.

[35] S. Liu, Z. Hao, C. Ying, H. Su, J. Zhu and Z. Cheng, A unified hard-constraint framework for solving geometrically complex PDEs, *Advances in Neural Information Processing Systems*, **35** (2022), 20287-20299.

[36] (MR4337511) [10.1137/21M1397908] 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 Journal on Scientific Computing*, **43** (2021), B1105-B1132.

[37] (MR4513793) [10.1016/j.jcp.2022.111722] L. D. McClenny and U. M. Braga-Neto, *Self-adaptive physics-informed neural networks*, *Journal of Computational Physics*, **474** (2023), 111722.

[38] (MR840029) [10.1016/0167-2789(86)90055-2] D. Michelson, *Steady solutions of the Kuramoto-Sivashinsky equation*, *Physica D: Nonlinear Phenomena*, **19** (1986), 89-111.

[39] [10.3934/Math.2017.2.479] A. Miranville, *The Cahn-Hilliard equation and some of its variants*, *AIMS Mathematics*, **2** (2017), 479-544.

[40] [10.1016/S0167-2789(99)00010-X] Y. Nishiura and D. Ueyama, *A skeleton structure of self-replicating dynamics*, *Physica D: Nonlinear Phenomena*, **130** (1999), 73-104.

[41] [10.1016/j.commatsci.2020.109687] M. T. Rad, A. Viardin, G. J. Schmitz and M. Apel, *Theory-training deep neural networks for an alloy solidification benchmark problem*, *Computational Materials Science*, **180** (2020), 109687.

[42] (MR3759415) [10.1016/j.jcp.2017.11.039] M. Raissi and G. E. Karniadakis, *Hidden physics models: Machine learning of nonlinear partial differential equations*, *Journal of Computational Physics*, **357** (2018), 125-141.

[43] (MR3881695) [10.1016/j.jcp.2018.10.045] 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*, *Journal of Computational Physics*, **378** (2019), 686-707.

[44] (MR2679727) J. Shen and X. Yang, Numerical approximations of Allen-Cahn and Cahn-Hilliard equations, *Discrete and Continuous Dynamical Systems*, **28** (2010), 1669-1691.

[45] Y. Shin, J. Darbon and G. E. Karniadakis, On the convergence and generalization of physics informed neural networks, preprint, arXiv:2004.01806, 2020.

[46] (MR4366261) [10.1016/j.cma.2021.114333] N. Sukumar and A. Srivastava, *Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks*, *Computer Methods in Applied Mechanics and Engineering*, **389** (2022), 114333.

[47] [10.1103/PhysRevE.91.032117] S. C. Takatori and J. F. Brady, *Towards a thermodynamics of active matter*, *Physical Review E*, **91** (2015), 032117.

[48] [10.1002/adfm.201501041] W. Tian, X. Mao, P. Brown, G. C. Rutledge and T. A. Hatton, *Electrochemically nanostructured polyvinylferrocene/polypyrrole hybrids with synergy for energy storage*, *Advanced Functional Materials*, **25** (2015), 4803-4813.

[49] (MR2731357) E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*, Springer Science & Business Media, 2013.

[50] R. Wang, M. Zhong, K. Xu, L. G. Sánchez-Cortés and I. de Cominges Guerra, PINNs-based uncertainty quantification for transient stability analysis, preprint, arXiv:2311.12947, 2023.

[51] (MR4701749) [10.1016/j.cma.2024.116813] S. Wang, S. Sankaran and P. Perdikaris, *Respecting causality for training physics-informed neural networks*, *Computer Methods in Applied Mechanics and Engineering*, **421** (2024), 116813.

[52] (MR4309866) [10.1137/20M1318043] S. Wang, Y. Teng and P. Perdikaris, *Understanding and mitigating gradient flow pathologies in physics-informed neural networks*, *SIAM Journal on Scientific Computing*, **43** (2021), A3055-A3081.

[53] (MR4337814) [10.1016/j.jcp.2021.110768] S. Wang, X. Yu and P. Perdikaris, *When and why PINNs fail to train: A neural tangent kernel perspective*, *Journal of Computational Physics*, **449** (2022), 110768.

[54] (MR483954) G. B. Whitham, *Linear and Nonlinear Waves*, Pure Appl. Math. Wiley-Interscience, New York-London-Sydney, 1974.

[55] (MR4203116) [10.4208/cicp.OA-2020-0086] C. L. Wight and J. Zhao, *Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks*, *Communications in Computational Physics*, **29** (2021), 930-954.

[56] (MR3975717) [10.1016/j.jcp.2019.06.041] X. Yang, D. Barajas-Solano, G. Tartakovsky and A. M. Tartakovsky, *Physics-informed cokriging: A Gaussian-process-regression-based multi-fidelity method for data-model convergence*, *Journal of Computational Physics*, **395** (2019), 410-431.

[57] [10.4249/scholarpedia.1435] A. M. Zhabotinsky, *Belousov-Zhabotinsky reaction*, *Scholarpedia*, **2** (2007), 1435.

[58] M. Zhong, D. Liu, R. Arroyave and U. Braga-Neto, Label propagation training schemes for physics-informed neural networks and Gaussian processes, preprint, arXiv:2404.05817, 2024.

[59] (MR4222510) [10.1007/s00466-020-01952-9] Q. Zhu, Z. Liu and J. Yan, *Machine learning for metal additive manufacturing: Predicting temperature and melt pool fluid dynamics using physics-informed neural networks*, *Computational Mechanics*, **67** (2021), 619-635.

Ablation Settings					Performance	
PBC	**IC1**	**IC2**	**SA**	**mini-batch**	**Rel. L^2 error**: *real part*	**Rel. L^2 error**: *imag part*
✓	✓	✗	✓	✓	1.80×10^{-4}	2.58×10^{-4}
✓	✗	✓	✓	✓	1.84×10^{-4}	3.27×10^{-4}
✓	✓	✗	✗	✓	7.34×10^{-3}	1.23×10^{-2}
✓	✗	✓	✗	✓	8.32×10^{-3}	1.39×10^{-2}
✓	✓	✗	✓	✗	1.10×10^{-3}	2.10×10^{-3}
✓	✗	✓	✓	✗	1.70×10^{-3}	3.27×10^{-3}
✓	✓	✗	✗	✗	1.00×10^{-1}	1.83×10^{-1}
✓	✗	✓	✗	✗	9.56×10^{-2}	1.56×10^{-1}
✓	✗	✗	✗	✗	8.09×10^{-1}	9.99×10^{-1}
✗	✓	✗	✗	✗	9.99×10^{-1}	8.78×10^{-1}
✗	✗	✓	✗	✗	5.04×10^{-2}	1.21×10^{-1}
✗	✗	✗	✓	✗	5.44×10^{-2}	7.26×10^{-2}
✗	✗	✗	✗	✓	6.60×10^{-2}	1.05×10^{-1}
✗	✗	✗	✗	✗	3.63×10^{-1}	5.06×10^{-1}

Table 5: *Nonlinear Schrödinger Equation:* Relative L^2 errors of real and imaginary equations for an ablation study illustrating the impact of disabling individual components of the proposed technique and training pipeline.

Solver	c	Rel. L^1 error	Rel. L^2 error
	30	0.0030	0.0041
HC-PINN	40	0.0141	0.0159
	50	0.0147	0.0164
	60	0.0171	0.0182
	30	0.0478	0.0579
CINN[4]	40	0.0714	0.0852
	50	0.4692	0.5365
	60	0.7293	0.7134
	30	0.0873	0.1003
Baseline	40	0.3918	0.4395
PINN	50	0.7140	0.7797
	60	0.8339	0.8664

Table 6: Comparison of HC-PINN, CINN and baseline PINN across different advection speeds c over 10 independent repetitions.

Ablation Settings			Performance	
PBC	**IC**	**SA**	**Rel. L^2 error**	**Rel. L^1 error**
✓	✓	✓	1.82×10^{-2}	1.71×10^{-2}
✓	✓	✗	9.75×10^{-2}	8.65×10^{-2}
✓	✗	✓	7.50×10^{-1}	7.87×10^{-1}
✗	✓	✓	3.87×10^{-2}	5.47×10^{-2}
✓	✗	✗	7.44×10^{-1}	8.01×10^{-1}
✗	✓	✗	1.07×10^{-1}	1.19×10^{-1}
✗	✗	✓	7.65×10^{-1}	7.71×10^{-1}
✗	✗	✗	8.66×10^{-1}	8.34×10^{-1}

Table 7: *Linear Advection Equation (c=60):* Relative L^2 and L^1 errors for an ablation study illustrating the impact of disabling individual components of the proposed technique and training pipeline.