

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

Baoli Hao^a, Ulisses Braga-Neto^b, Chun Liu^a, Lifan Wang^c, Ming Zhong^d

^a Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, USA

^b Department of Electrical and Computing Engineering, Texas A&M University, College Station, TX, USA

^c Department of Physics and Astrophysics, Texas A&M University, College Station, TX, USA

^d Department of Mathematics, University of Houston, TX, USA

October 9, 2025

Abstract

Training Physics-Informed Neural Networks (PINNs) on stiff time-dependent PDEs remains highly unstable. Through rigorous ablation studies, we identify a surprisingly critical factor: the enforcement of initial conditions. We present the first systematic ablation of two core strategies, hard initial-condition constraints and adaptive loss weighting. Across challenging benchmarks (sharp transitions, higher-order derivatives, coupled systems, and high frequency modes), we find that exact enforcement of initial conditions (ICs) is not optional but essential. Our study demonstrates that stability and efficiency in PINN training fundamentally depend on ICs, paving the way toward more reliable PINN solvers in stiff regimes.

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 (Allen and Cahn, 1979; Bazant, 2017; Burgers, 1948; Cahn and Hilliard, 1958; Cassani, Monteverde, and Piumetti, 2021; De Kepper, Castets, Dulos, and Boissonade, 1991; Fibich, 2015; Horstmann, Danner, and Bessler, 2013; Hyman, Weber, and Jülicher, 2014; Kato, 1987; Kevrekidis, Rasmussen, and Bishop, 2001; Kudryashov, 1990; Lee, Huh, Jeong, Shin, Yun, and Kim, 2014; Michelson, 1986; Miranville, 2017; Nishiura and Ueyama, 1999; Shen and Yang, 2010; Takiatori and Brady, 2015; Tian, Mao, Brown, Rutledge, and Hatton, 2015; Whitham, 2011; Zhabotinsky, 2007). However, solution of stiff time-dependent PDEs can be exceedingly challenging to obtain and often requires substantial computational resources and specialized numerical techniques.

Machine learning methodologies have demonstrated remarkable success across various scientific and engineering fields, transforming areas such as protein folding prediction (Jumper, Evans, Pritzel, Green, Figurnov, Ronneberger, Tunyasuvunakool, Bates, Žídek, Potapenko, et al., 2021), drug discovery and development (Carracedo-Reboreda, Liñares-Blanco, Rodríguez-Fernández, Cedrón, Novoa, Carballal, Maojo, Pazos, and Fernandez-Lozano, 2021), and even the optimization of fundamental algorithmic processes such as matrix-vector multiplication (Fawzi, Balog, Huang, Hubert, Romera-Paredes, Barekatian, Novikov, R. Ruiz, Schrittweise, Swirszcz, et al., 2022). 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 (Cai, Wang, Lu, Zaki, and Karniadakis, 2021; Chen, Jeffery, Zhong, McClelly, Braga-Neto, and Wang, 2022; Chen, Hosseini, Owhadi, and Stuart, 2021; Jin, Cai, Li, and Karniadakis, 2021; Liu and Wang, 2019, 2020; Rad, Viardin, Schmitz, and Apel, 2020; Raissi

and Karniadakis, 2018; Raissi, Perdikaris, and Karniadakis, 2019; Wang, Zhong, Xu, Sánchez-Cortés, and Guerra, 2023; Yang, Barajas-Solano, Tartakovsky, and Tartakovsky, 2019; Zhu, Liu, and Yan, 2021). 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 (Coutinho, Dall'Aqua, McClenney, Zhong, Braga-Neto, and Gildin, 2023; McClenney and Braga-Neto, 2020; Wang, Teng, and Perdikaris, 2020a; Wang, Yu, and Perdikaris, 2020b).

This paper presents a complete study of investigating the importance of the training components for PINNs on solving stiff time-dependent PDES. First, we conduct an ablation study of training algorithms for PINNs applied to stiff time-dependent PDEs, focusing specifically on two schemes: hard constraints and self-adaptive loss weights (McClenney & Braga-Neto, 2020). The hard-constraint scheme directly embeds information about initial and boundary conditions into the neural network architecture. This transformation reduces the effects of spectral bias from training a multi-objective loss, leading to improved solution accuracy (Wang et al., 2020b). In addition, we investigate the use of adaptive loss weights to alleviate spectral bias in PINN training (McClenney & Braga-Neto, 2020). Through our ablation study of these training schemes, we conclude that the hard-constraint transformation provides the best improvement. Next, we provide a Neural Tangent Kernel perspective on how the hard-constrained transformation is reducing the spectral bias in PINN training. We emphasize that hard constraints is not the only effective strategies for handling stiff PDEs. Such method can be integrated with other advanced PINN techniques, such as causal PINNs (Wang et al., 2020b), time-marching PINNs (Wight and Zhao, 2020), residual-based-attention (RBA) PINNs (Anagnostopoulos, Toscano, Stergiopoulos, and Karniadakis, 2024), curriculum training (Krishnapriyan, Gholami, Zhe, Kirby, and Mahoney, 2021), and co-training PINNs (Zhong, Liu, Arroyave, and Braga-Neto, 2024), to achieve synergistic improvements in prediction 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 presented a detailed discussion on the theoretical mechanism behind the hard-constrained transformation in section 3.1; we presented the results (in summary) from four major categories of examples in section 4; we conclude our paper in 5, pointing out several future directions for this line of research. Since the examples we conducted are too extensive, the additional details are presented in Appendix section B.

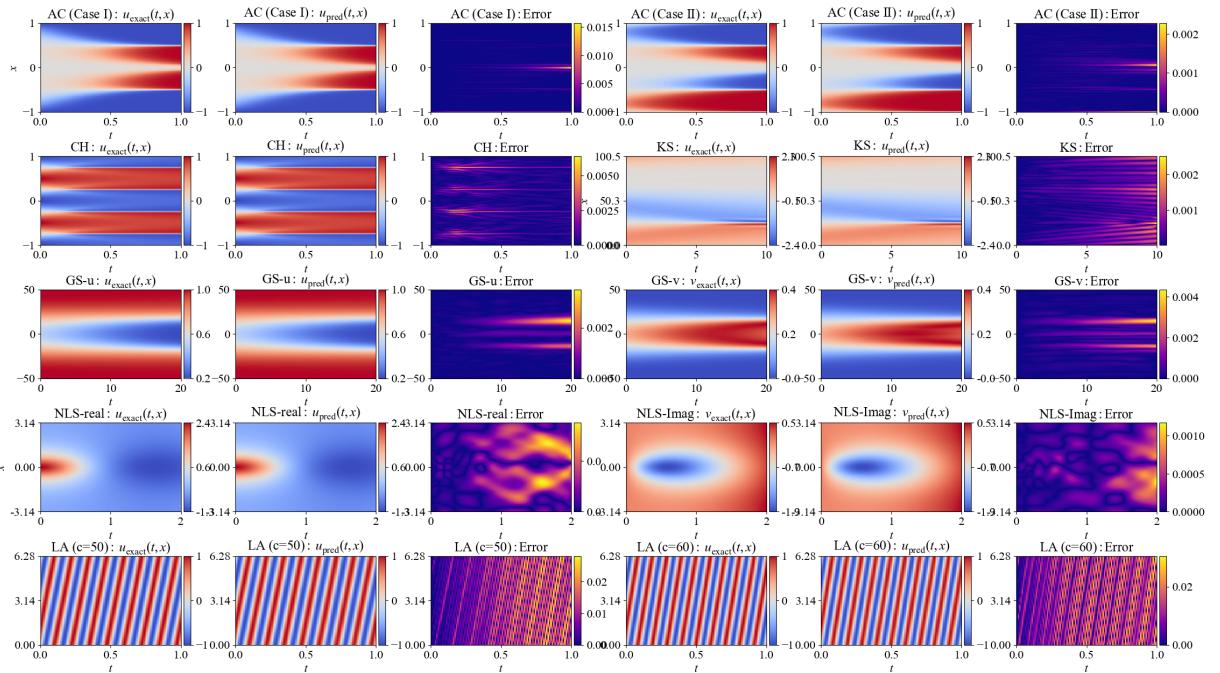


Figure 1: 7 Benchmark Results: Truth vs HC-PINN vs Absolute Error.

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 (Shin, Darbon, and Karniadakis, 2020; Wang et al., 2020a,2; Wight & Zhao, 2020). This imbalance often leads gradient descent to prioritize minimizing the residual loss over 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 (Haitsiukovich and Ilin, 2022; Krishnapriyan et al., 2021; McClenny & Braga-Neto, 2020; Wang, Sankaran, and Perdikaris, 2022; Wight & Zhao, 2020).

Several strategies have been proposed to address these challenges. One approach, introduced by Wight & Zhao (2020), 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 (McClenny & Braga-Neto, 2020; Wight & Zhao, 2020). McClenny & Braga-Neto (2020) 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 (Krishnapriyan et al., 2021), causality PINN (Wang et al., 2022), artificial viscosity PINN (Coutinho et al., 2023), Co-training PINN (Zhong et al., 2024).

Beyond loss weighting and time marching, structural modifications to PINNs have also been explored. Braga-Neto (2022) 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 (Lagaris, Likas, and Fotiadis, 1998). Our method builds upon these architectural modifications, with a focus on enforcing (near)-exact fit of the initial condition using hard-constrained transformation.

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 $\mathbf{x} = (x_1 \dots x_d)^\top \in \Omega$. Then we say that u is a solution of a time-dependent PDE, if u satisfies the following

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

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

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

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

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

Furthermore \mathcal{B} is an operator defined for (t, \mathbf{x}) on the boundary. We also assume that the functions u_0 , f , and g are user inputs that satisfy the desired regularity so that the existence and uniqueness of solutions for such a PDE are guaranteed. We also assume that the compatibility condition, $g(0, \mathbf{x}) = u_0(\mathbf{x})$ for $\mathbf{x} \in \partial\Omega$, is satisfied to prevent any ill-conditioning of the solution.

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,

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

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

$$\mathbf{e}_i = \underbrace{[0 \quad \cdots \quad 0 \quad 1 \quad 0 \quad \cdots \quad 0]}_{1 \text{ is the in the } i^{th} \text{ position.}}^\top.$$

One can even consider a mixture of two or all of the above BC conditions. For example, we can have $\Omega = \Gamma_1 \cup \dots \cup \Gamma_K$ with each Γ_i defines a different kind of BC. 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}_{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

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

where λ is a regularization parameter, and the two losses are as follows

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

for $u_{nn} \in \mathcal{H}_{NN}$. 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 us to consider hard constraints 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 (McClenny & Braga-Neto, 2020). 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

$$\tilde{u}(t, \mathbf{x}) = \psi(t, \mathbf{x}) + \phi(t, \mathbf{x})u_{nn}(t, \mathbf{x}), \quad (3)$$

Here the functions ψ and ϕ are smooth functions with the following property

$$\left\{ \begin{array}{ll} \psi(0, \mathbf{x}) = u_0(\mathbf{x}), & \psi(0, \mathbf{x}) = 0, \quad \text{for } \mathbf{x} \in \bar{\Omega}, \\ \psi(t, \mathbf{x}) = g(t, \mathbf{x}), & \psi(t, \mathbf{x}) = 0, \quad \text{for } (t, \mathbf{x}) \in [0, T] \times \partial\Omega. \end{array} \right. \quad (4)$$

In the case of periodic BC, we simply require ψ and ϕ to be both P -periodic. When such transformation is used, then the training for u^{nn} is updated as

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

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 $v(t, x) = [t \quad \mathcal{F}_m(x)]^\top$ with

$$\mathcal{F}_m(x) = [1 \quad \cos\left(\frac{2\pi}{P}x\right) \quad \sin\left(\frac{2\pi}{P}x\right) \quad \cdots \quad \cos\left(\frac{2\pi}{P}mx\right) \quad \sin\left(\frac{2\pi}{P}mx\right)]$$

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 (Dong and Ni, 2021). For $\mathbf{x} = [x \quad y]$, we can use the following

$$v(t, x, y) = [t \quad \mathcal{F}_m(x) \quad \mathcal{F}_m(y)];$$

similarly for higher dimensional $\mathbf{x} = [x_1 \quad \cdots \quad x_d]$, we have

$$v(t, \mathbf{x}) = [t \quad \mathcal{F}_m(x_1) \quad \cdots \quad \mathcal{F}_m(x_d)].$$

2.2 Other Training Enhancements

We also consider mini-batching (to handle large sample size) and self-adaptive loss weights (McClenny & Braga-Neto, 2020), please see sections A.1 and A.2 for details.

3 Theoretical Foundation for Hard Constraints

We discuss the well-conditioning condition for the hard-constraint transformation and the Neural Tangent Kernel (NTK) formulation for the training decay rates in this section.

3.1 Well Condition of the Transformation

We start with the minimal regularity conditions on the pair (ψ, ϕ) for a well-defined transformation.

Theorem 1 (Well Conditioning). *The transformation is well defined when ϕ and ψ is C^1 in time and have K^{th} order partial derivatives w.r.t \mathbf{x} . Moreover, $\phi_t(0, \mathbf{x}) \neq 0$. Then u_{nn} will satisfy a new PDE*

$$\begin{cases} \phi \partial_t u_{nn} + \mathcal{P}[\psi + \phi u_{nn}] + \phi_t u_{nn} = f - \psi_t, & (t, \mathbf{x}) \in (0, T] \times \Omega, \\ u_{nn}(0, \mathbf{x}) = \frac{u_t(0, \mathbf{x}) - \psi_t(0, \mathbf{x})}{\phi_t(0, \mathbf{x})}, & \mathbf{x} \in \bar{\Omega}. \end{cases}$$

Remark 1. Hence our choice of setting $\psi(t, \mathbf{x}) = e^{-Ct} u_0(\mathbf{x})$ and $\phi(t, \mathbf{x}) = t$ satisfies the well-conditioning requirement. However, if we are interested in obtaining the solution at $T \rightarrow \infty$, using $\phi(t, \mathbf{x}) = t$ will cause the learning to deteriorate, since $t \rightarrow \infty$. It is better to use $\phi(t, \mathbf{x}) = 1 - e^{-Ct}$ with $\psi(t, \mathbf{x}) = e^{-Ct} u_0(\mathbf{x})$, so that the influence of ϕ on u_{nn} will settle down to a fixed number. Therefore we will obtain (when $T \rightarrow \infty$)

$$u_{nn}(\infty, \mathbf{x}) = \mathcal{P}^{-1}[f](\infty, \mathbf{x}) = u(\infty, \mathbf{x}).$$

In the case of \mathcal{P} being linear and $\phi(t, \mathbf{x}) = \phi(t)$, we end up with a much simpler PDE

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

If $\phi(t) \neq 0$ when $t \neq 0$, then it can be simplified further to

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

We can see that u_{nn} satisfies a similar PDE to the original one with the additional reaction term $\frac{\phi'}{\phi} u_{nn}$ and updated forcing $\frac{f - \psi_t - \mathcal{P}(\psi)}{\phi}$, which contains the information of u_0 and g .

3.2 Why Initial Conditions Matter?

The hard constraints transformation consider both the boundary condition and initial condition, however our emphasis is on the initial condition as the focus of our paper is about solving stiff time-dependent PDEs with PINNs. Stiff time-dependent PDEs have important features that exhibit over a widely varying time scaling, hence making even explicit time-integration methods from traditional numerical methods difficult to capture these features. Therefore by removing the initial condition through hard constraint transformation, we achieve the following superior performance

- We implicitly enforce the time direction into the training of PINNs, reducing the spectral bias from training with multi-objective loss.
- We mimic the behavior of traditional numerical methods, as they start out with either exact or approximated initial condition in their time integration.
- The hard constraint transformation works similarly like an implicit time integrator.

3.3 How Hard Constraints Work?

Theorem 2. *Assume that \mathcal{P} is linear and $\phi(t, \mathbf{x}) = \phi(t) \neq 0$ for $t > 0$, then for a one-hidden layer network, we have for a network $u(\mathbf{y}; \boldsymbol{\theta})$ with $\boldsymbol{\theta} \in \mathbb{R}^D$ being the network parameters, the following relationship is satisfied*

$$u(\mathbf{y}; \boldsymbol{\theta}) = \mathbf{h}^\top(\mathbf{y})\boldsymbol{\theta}, \quad \mathbf{h} = [h_1(\mathbf{y}) \quad \cdots \quad h_D(\mathbf{y})]^\top,$$

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

$$\begin{aligned} Loss_s(u(\mathbf{y}; \boldsymbol{\theta})) &= \frac{1}{2N}|\mathbf{A}\boldsymbol{\theta} - \mathbf{f}|_{\ell^2(\mathbb{R}^N)}^2 + \frac{1}{2M}|\mathbf{C}\boldsymbol{\theta} - \mathbf{u}_0|_{\ell^2(\mathbb{R}^M)}^2, \quad (\text{soft constrained}) \\ Loss_h(u(\mathbf{z}; \boldsymbol{\theta})) &= \frac{1}{2N}|\Lambda_\phi \mathbf{A}\boldsymbol{\theta} + \Lambda_{\phi_t} \mathbf{B}\boldsymbol{\theta} - \tilde{\mathbf{f}}|_{\ell^2(\mathbb{R}^N)}^2, \quad (\text{hard constrained}) \end{aligned}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathcal{L}_s[\mathbf{h}^\top(\mathbf{z}_1)] \\ \vdots \\ \mathcal{L}_s[\mathbf{h}^\top(\mathbf{z}_N)] \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{h}^\top(\mathbf{z}_1) \\ \vdots \\ \mathbf{h}^\top(\mathbf{z}_N) \end{bmatrix}, \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{h}^\top(\zeta_1) \\ \vdots \\ \mathbf{h}^\top(\zeta_M) \end{bmatrix},$$

where $\mathbf{x}_i^{IC} \in \bar{\Omega}$; and for the diagonal matrices Λ_ϕ and Λ_{ϕ_t}

$$\Lambda_\phi = diag(\phi(\mathbf{z}_1), \dots, \phi(\mathbf{z}_N)) \quad \text{and} \quad \Lambda_{\phi_t} = diag(\phi_t(\mathbf{z}_1), \dots, \phi_t(\mathbf{z}_N)),$$

and the vectors

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

Therefore the Hessians of the losses are

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

Then the Hessians are bounded as

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

where $|\mathcal{L}_s|_2$ is the L^2 -operator norm of \mathcal{L}_s and

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

Remark 2. Notice the competition between the PDE residual points and the initial conditions points (reflected in \mathbf{B} and \mathbf{C}) in the bound for H_s , where for H_h , it is all about the operator \mathcal{L}_s and the scaling by ϕ and ϕ_t . The bounds will become much more complex when \mathcal{P} is not linear or ϕ also depends on \mathbf{x} .

In order to give a better picture on the training decay rates, we provide a Neural Tangent Kernel (NTK) perspective on how hard constraints would work for 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 $\{\zeta_j = (0, \mathbf{x}_j)\}_{j=1}^M$ 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

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

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

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

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

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

with $\boldsymbol{\theta}_s(0) = \boldsymbol{\theta}_h(0) = \boldsymbol{\theta}_0$. Let the operators $\mathcal{L}_s[u] := \partial_t u + \mathcal{P}[u]$ and $\mathcal{L}_h[u] := \phi \partial_t u + \mathcal{P}[\psi + \phi u] + \phi u$. Then with NTK, we obtain

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

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

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

Meanwhile, for u_h , we have

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

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 $\mathbf{K}^s(\tau) \approx \mathbf{K}^s(0) = \mathbf{K}_*^s$ and $\mathbf{K}^h(\tau) \approx \mathbf{K}^h(0) = \mathbf{K}_*^h$, then the decays satisfy

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

where $\mathbf{K}_*^s = \mathbf{Q}_s^\top \mathbf{\Lambda}_s \mathbf{Q}_s$; moreover

$$\frac{d\mathcal{L}_h[u_h](\mathbf{z}; \boldsymbol{\theta}_h(\tau))}{d\tau} - (f(\mathbf{z}) + \psi_t(\mathbf{z})) = -e^{\mathbf{K}_*^h \tau} (f(\mathbf{z}) + \psi_t(\mathbf{z})) = -\mathbf{Q}_h^\top e^{-\mathbf{\Lambda}_h \tau} \mathbf{Q}_h (f(\mathbf{z}) + \psi_t(\mathbf{z})),$$

where $\mathbf{K}_*^h = \mathbf{Q}_h^\top \mathbf{\Lambda}_h \mathbf{Q}_h$. One can see that the training decay of not using the hard-constrained transformation depends on both the eigenvalues of \mathbf{K}_{11}^s and \mathbf{K}_{22}^s ; whereas the training decay of the hard-constrained transformation depends only on \mathbf{K}^h , hence reducing the spectral bias.

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 and flexibility in addressing a broad class of challenging problems. These equations often involve strong nonlinearities, stiffness, and parameter sensitivities, posing significant challenges for data-driven methods such as PINNs. Nonetheless, our results illustrate the capacity of the proposed framework to overcome these difficulties and accurately approximate the solutions across these varied regimes.

To begin, we employ Latin hypercube sampling to generate collocation points across the full spatial-temporal domain, including its boundary. To facilitate training and better incorporate the initial condition into the solution structure, we adopt the following transformation:

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

where $u_0(\mathbf{x})$ denotes the initial condition, and we test two values of $C \in \{0.1, 1\}$, to assess how the decay rate affects learned dynamics. The transformed PINN u_{nn} is trained to minimize a loss function that accounts for the PDE residuals, which now includes IC/BC.

We present the results in four major categories, (*I*) ablation study across 7 different stiff PDEs where stiffness comes in different forms, such as sharp phase transition in Allen-Cahn, higher-order derivatives (forth order) in Cahn-Hilliard and Kuramoto-Sivashinsky, coupled outputs form Gray-Scott and Shrödinger systems, and high frequency modes in fast moving linear transport; (*II*) comparison to other start-of-the-art PINNs such as the Causal PINN ([Wang et al., 2022](#)) and Enhanced RBA PINN ([Anagnostopoulos et al., 2024](#)) on Allen-Cahn, Cahn-Hilliard and Kuramoto-Sivashinsky; (*III*) application of HC-PINNs on a 2D Allen-Cahn; (*IV*) sensitivity analysis on the parameters m for the Allen-Cahn equation with two different initial conditions.

	Causal training (MLP)	Enhanced RBA	HC-PINNs
AC (Case I)	6.95×10^{-2}	2.62×10^{-3}	8.29×10^{-4}
AC (Case II)	1.78×10^{-2}	1.08×10^{-3}	2.14×10^{-4}
CH	3.49×10^{-1}	9.83×10^{-2}	5.61×10^{-4}
KS	2.72×10^{-1}	3.64×10^{-2}	5.37×10^{-4}

Table 1: Relative L^2 errors for the Allen–Cahn (AC), Cahn–Hilliard (CH) and Kuramoto–Sivashinsky (KS) equations obtained after $50k$ training iterations. All methods are implemented under the same $50k$ iteration budget for a fair comparison. Note that the reported errors for Causal training ([Wang et al., 2022](#)) and Enhanced RBA ([Anagnostopoulos et al., 2024](#)) differ from those in their original papers, where results were obtained with $300k$ iterations.

Figure 1 provides a summary of our ablation study over 7 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, nonlinear Schrödinger, and linear advection equations (with different speeds $c = 50, 60$). Complementing these visual comparisons, Table 1 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 for $50k$ iterations, same data and parameter settings to ensure fairness. Notably, while prior works originally reported results with $300k$ iterations, HC-PINNs achieve substantially lower errors even with a much shorter training 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. For full experimental details and visualizations, see Section B in the appendix.

The results of solving the 2D Allen–Cahn equation in Figure 2 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. 5, 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.

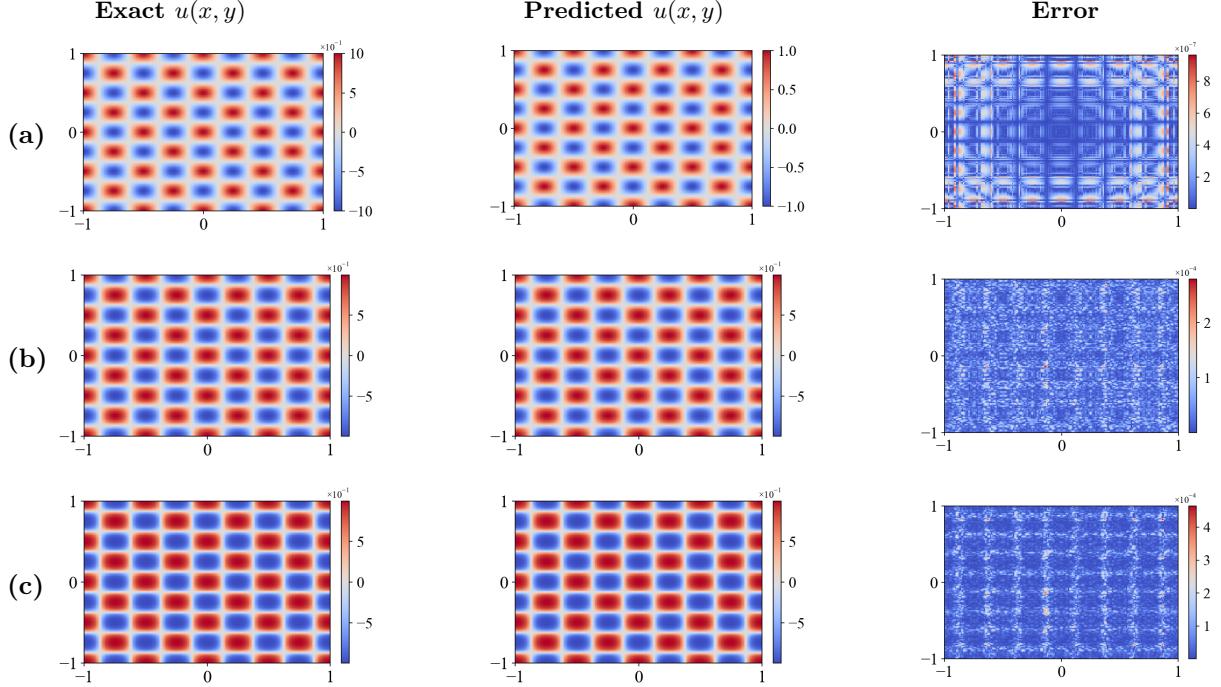


Figure 2: Reference, HC-PINN predicted solution, and absolute error of the 2D Allen Cahn equation at different time snapshots (a) $t = 0$, (b) $t = 0.5$, (c) $t = 1$.

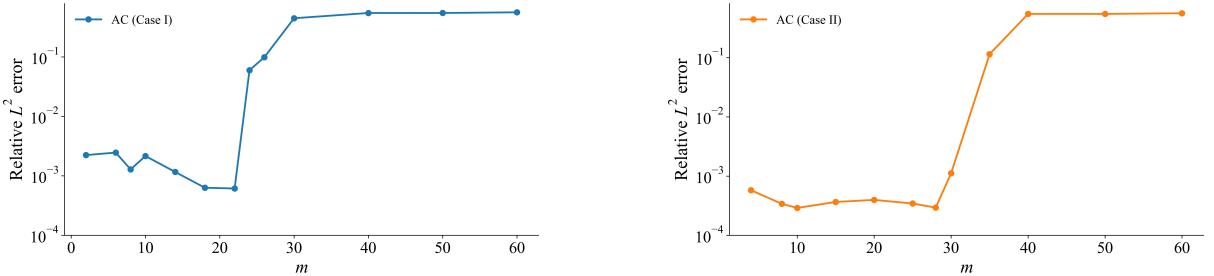


Figure 3: Sensitivity of HC-PINNs to the number of Fourier modes m in the Allen–Cahn equations. The plot shows the relative L^2 error as a function of m .

5 Conclusion

We provide a comprehensive study of hard-constraints and self-adaptive loss weights to study the stabilization of PINN training on solving stiff time-dependent PDEs. Our comprehensive study includes an ablation study of seven distinct types of PDES, comparison study, a theoretical analysis, an application to 2D Allen Chan, and a sensitive analysis of m (the number of Fourier modes).

From the ablation study, we found out that the transformation provided by hard constraint significantly alleviates the training challenges typically associated with stiff time-dependent PDEs. Our ablation study demonstrates effectiveness across seven distinct types of such PDEs, encompassing scalar-valued and vector-valued (including complex-valued) outputs. This approach not only enhances training efficiency but also improves the accuracy and stability of the learned solutions, particularly in the presence of stiffness, where conventional PINNs often struggle. We provide theoretical insights using NTK into how hard-constrained transformation is reducing the training difficulties, especially in terms of reducing the spectral bias. The comparison study focuses on comparing our HC-PINN to two other state-of-the-art PINN variants on established examples.

Future work includes extending the hard-constraint framework to more two- and three-dimensional PDEs, which pose additional challenges in both architecture design and computational cost. Ongoing research also focuses on identifying optimal transformations ψ and ϕ , which govern the embedding of IC and BC into the network structure, with the goal of further enhancing performance and generalizability.

References

- Samuel M Allen and John W Cahn. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta metallurgica*, 27(6):1085–1095, 1979.
- Sokratis J Anagnostopoulos, Juan Diego Toscano, Nikolaos Stergiopoulos, and George Em Karniadakis. Residual-based attention in physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116805, 2024.
- Martin Z Bazant. Thermodynamic stability of driven open systems and control of phase separation by electro-autocatalysis. *Faraday discussions*, 199:423–463, 2017.
- Ulisses Braga-Neto. Characteristics-informed neural networks for forward and inverse hyperbolic problems. *arXiv preprint arXiv:2212.14012*, 2022.
- Johannes Martinus Burgers. A mathematical model illustrating the theory of turbulence. *Advances in applied mechanics*, 1:171–199, 1948.
- John W Cahn and John E Hilliard. Free energy of a nonuniform system. i. interfacial free energy. *The Journal of chemical physics*, 28(2):258–267, 1958.
- Shengze Cai, Zhicheng Wang, Lu Lu, Tamer A Zaki, and George Em Karniadakis. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436:110296, 2021.
- Paula Carracedo-Reboredo, Jose Liñares-Blanco, Nereida Rodríguez-Fernández, Francisco Cedrán, Francisco J Novoa, Adrian Carballal, Victor Maojo, Alejandro Pazos, and Carlos Fernandez-Lozano. A review on machine learning approaches and trends in drug discovery. *Computational and structural biotechnology journal*, 19:4538–4558, 2021.
- Andrea Cassani, Alessandro Monteverde, and Marco Piumetti. Belousov-zhabotinsky type reactions: The non-linear behavior of chemical systems. *Journal of mathematical chemistry*, 59:792–826, 2021.
- Xingzhuo Chen, David J Jeffery, Ming Zhong, Levi McClelly, Ulisses Braga-Neto, and Lifan Wang. Using physics informed neural networks for supernova radiative transfer simulation. *arXiv preprint arXiv:2211.05219*, 2022.
- Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Solving and learning nonlinear pdes with gaussian processes. *Journal of Computational Physics*, 447:110668, 2021.
- Emilio Jose Rocha Coutinho, Marcelo Dall'Aqua, Levi McClelly, Ming Zhong, Ulisses Braga-Neto, and Eduardo Gildin. Physics-informed neural networks with adaptive localized artificial viscosity. *Journal of Computational Physics*, 489:112265, 2023.
- 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(1-2):161–169, 1991.
- Suchuan Dong and Naxian Ni. A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *Journal of Computational Physics*, 435:110242, 2021.
- Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatian, Alexander Novikov, Francisco J R. Ruiz, Julian Schrittweiser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.
- Gadi Fibich. *The nonlinear Schrödinger equation*, volume 192. Springer, 2015.
- P Gray and SK Scott. Autocatalytic reactions in the isothermal, continuous stirred tank reactor: isolas and other forms of multistability. *Chemical Engineering Science*, 38(1):29–43, 1983.
- Peter Gray and Stephen K Scott. Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Oscillations and instabilities in the system $a + 2b \rightarrow 3b; b \rightarrow c$. *Chemical Engineering Science*, 39(6):1087–1097, 1984.
- Katsiaryna Haitsiukevich and Alexander Ilin. Improved training of physics-informed neural networks with model ensembles. *arXiv preprint arXiv:2204.05108*, 2022.
- Birger Horstmann, Timo Danner, and Wolfgang G Bessler. Precipitation in aqueous lithium–oxygen batteries: a model-based analysis. *Energy & Environmental Science*, 6(4):1299–1314, 2013.
- Anthony A Hyman, Christoph A Weber, and Frank Jülicher. Liquid-liquid phase separation in biology. *Annual review of cell and developmental biology*, 30:39–58, 2014.
- Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, 2021.
- John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate

- protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.
- Tosio Kato. On nonlinear schrödinger equations. In *Annales de l'IHP Physique théorique*, volume 46, pp. 113–129, 1987.
- PG Kevrekidis, KØ Rasmussen, and AR Bishop. The discrete nonlinear schrödinger equation: a survey of recent results. *International Journal of Modern Physics B*, 15(21):2833–2900, 2001.
- Junseok Kim, Seunggyu Lee, Yongho Choi, Seok-Min Lee, Darae Jeong, et al. Basic principles and practical applications of the cahn–hilliard equation. *Mathematical Problems in Engineering*, 2016, 2016.
- Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.
- Nikolai A Kudryashov. Exact solutions of the generalized kuramoto-sivashinsky equation. *Physics Letters A*, 147(5-6):287–291, 1990.
- Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.
- Dongsun Lee, Joo-Youl Huh, Darae Jeong, Jaemin Shin, Ana Yun, and Junseok Kim. Physical, mathematical, and numerical derivations of the cahn–hilliard equation. *Computational Materials Science*, 81:216–225, 2014.
- Randall J LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.
- Jiangyan Liang, Ning Jiang, Chun Liu, Yiwei Wang, and Teng-Fei Zhang. On a reversible gray-scott type system from energetic variational approach and its irreversible limit. *Journal of Differential Equations*, 309:427–454, 2022.
- Dehao Liu and Yan Wang. Multi-fidelity physics-constrained neural network and its application in materials modeling. *Journal of Mechanical Design*, 141(12):121403, 2019.
- Dehao Liu and Yan Wang. A dual-dimer method for training physics-constrained neural networks with minimax architecture. *Neural Networks*, 136:112–125, 2021.
- Levi McClenney and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.
- Daniel Michelson. Steady solutions of the kuramoto-sivashinsky equation. *Physica D: Nonlinear Phenomena*, 19(1):89–111, 1986.
- Alain Miranville. The cahn–hilliard equation and some of its variants. *AIMS Mathematics*, 2(3):479–544, 2017.
- Yasumasa Nishiura and Daishin Ueyama. A skeleton structure of self-replicating dynamics. *Physica D: Nonlinear Phenomena*, 130(1-2):73–104, 1999.
- M Torabi Rad, A Viardin, GJ Schmitz, and M Apel. Theory-training deep neural networks for an alloy solidification benchmark problem. *Computational Materials Science*, 180:109687, 2020.
- Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.
- Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.
- Jie Shen and Xiaofeng Yang. Numerical approximations of allen-cahn and cahn-hilliard equations. *Discrete Contin. Dyn. Syst.*, 28(4):1669–1691, 2010.
- Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the convergence and generalization of physics informed neural networks. *arXiv preprint arXiv:2004.01806*, 2020.
- Sho C Takatori and John F Brady. Towards a thermodynamics of active matter. *Physical Review E*, 91(3):032117, 2015.
- Wenda Tian, Xianwen Mao, Paul Brown, Gregory C Rutledge, and T Alan Hatton. Electrochemically nanostructured polyvinylferrocene/polypyrrole hybrids with synergy for energy storage. *Advanced Functional Materials*, 25(30):4803–4813, 2015.
- Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.
- Ren Wang, Ming Zhong, Kaidi Xu, Lola Giráldez Sánchez-Cortés, and Ignacio de Cominges Guerra. Pinns-based uncertainty quantification for transient stability analysis. *arXiv preprint arXiv:2311.12947*, 2023.
- Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536*, 2020a.

- Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *arXiv preprint arXiv:2007.14527*, 2020b.
- Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022.
- Gerald Beresford Whitham. *Linear and nonlinear waves*. John Wiley & Sons, 2011.
- Colby L Wight and Jia Zhao. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.
- Xiu Yang, David Barajas-Solano, Guzel Tartakovsky, and Alexandre M Tartakovsky. Physics-informed cokriging: A gaussian-process-regression-based multifidelity method for data-model convergence. *Journal of Computational Physics*, 395:410–431, 2019.
- Anatol M Zhabotinsky. Belousov-zhabotinsky reaction. *Scholarpedia*, 2(9):1435, 2007.
- Ming Zhong, Dehao Liu, Raymundo Arroyave, and Ulisses Braga-Neto. Label propagation training schemes for physics-informed neural networks and gaussian processes. *arXiv preprint arXiv:2404.05817*, 2024.
- Qiming Zhu, Zeliang Liu, and Jinhui Yan. Machine learning for metal additive manufacturing: predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Computational Mechanics*, 67(2):619–635, 2021.

A Methodology: Additional Details

In this section, we discuss the additional methods used in the ablation study, including mini-batching and self-adaptive loss weights.

A.1 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 (Wight & Zhao, 2020), the mini-batching technique is similar to time-marching resampling but without an explicit time grid.

A.2 Self-Adaptive PINNs

The proper choice of penalty term λ can be updated from derived formula as mentioned in (Wang et al., 2020b). However, data-driven discovery of λ is possible. Outline in (McClenny & Braga-Neto, 2020), the loss is further modified as

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

and

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

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 (McClenny & Braga-Neto, 2020) 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.

A.3 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 measure the discrepancy between the predicted and reference solutions in average (energy-based) and absolute (mass-based) senses, respectively. The relative form normalizes the error with respect to the reference solution, enabling 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 $\mathcal{U}(x_i, t_i)$.

A.4 Proofs of the Theorems

We present the proofs for the two aforementioned theorem.

Theorem (Well Conditioning). *The transformation is well defined when ϕ and ψ is C^1 in time and have K^{th} order partial derivatives w.r.t \mathbf{x} . Moreover, $\phi_t(0, \mathbf{x}) \neq 0$. Then u_{nn} will satisfy a new PDE*

$$\begin{cases} \phi \partial_t u_{nn} + \mathcal{P}[\psi + \phi u_{nn}] + \phi_t u_{nn} = f - \psi_t, & (t, \mathbf{x}) \in (0, T] \times \Omega, \\ u_{nn}(0, \mathbf{x}) = \frac{u_t(0, \mathbf{x}) - \psi_t(0, \mathbf{x})}{\phi_t(0, \mathbf{x})}, & \mathbf{x} \in \bar{\Omega}. \end{cases}$$

Proof. Since $\mathcal{L}_s[u] = \partial_t u + \mathcal{P}[u]$ is applied to the transformation $u = \psi + \phi u_{nn}$ and u_{nn} is constructed with hyper tangent as the activation function, we only need ψ and ϕ to satisfy the required regularity from \mathcal{L}_s so that the operator can still be applied, hence ψ and ϕ have to have first-order time derivative and K^{th} -order partial derivatives w.r.t to \mathbf{x} , we further require them to be continuous in order to allow classical solutions to exist for \mathcal{L}_s . Meanwhile from the transformation we have

$$u_{nn}(t, \mathbf{x}) = \frac{u(t, \mathbf{x}) - \psi(t, \mathbf{x})}{\phi(t, \mathbf{x})}, \quad \text{for } (t, \mathbf{x}) \in (0, T] \times \Omega.$$

As $t \rightarrow 0$, we have $\psi(0, \mathbf{x}) = u_0(\mathbf{x})$ and $\phi(0, \mathbf{x}) = 0$, hence by L'Hôpital's rule

$$\lim_{t \rightarrow 0} u_{nn}(t, \mathbf{x}) = \lim_{t \rightarrow 0} \frac{u(t, \mathbf{x}) - \psi(t, \mathbf{x})}{\phi(t, \mathbf{x})} = \lim_{t \rightarrow 0} \frac{u_t(t, \mathbf{x}) - \psi_t(t, \mathbf{x})}{\phi_t(t, \mathbf{x})},$$

when $\phi_t(0, \mathbf{x}) \neq 0$, then limit is defined and we can have

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

Moreover

$$\mathcal{L}_s(\psi + \phi u_{nn}) = \partial_t(\psi + \phi u_{nn}) + \mathcal{P}[\psi + \phi u_{nn}] = \phi \partial_t u_{nn} + \mathcal{P}[\psi + \phi u_{nn}] + \phi_t u_{nn} + \psi_t,$$

hence the new PDE for u_{nn} . □

Next, the theorem on Hessian bounds.

Theorem. *Assume that \mathcal{P} is linear and $\phi(t, \mathbf{x}) = \phi(t) \neq 0$ for $t > 0$, then for a one-hidden layer network, we have for a network $u(\mathbf{y}; \boldsymbol{\theta})$ with $\boldsymbol{\theta} \in \mathbb{R}^D$ being the network parameters, the following relationship is satisfied*

$$u(\mathbf{y}; \boldsymbol{\theta}) = \mathbf{h}^\top(\mathbf{y}) \boldsymbol{\theta}, \quad \mathbf{h} = [h_1(\mathbf{y}) \quad \dots \quad h_D(\mathbf{y})]^\top,$$

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

$$\text{Loss}_s(u(\mathbf{y}; \boldsymbol{\theta})) = \frac{1}{2N} \|\mathbf{A}\boldsymbol{\theta} - \mathbf{f}\|_{\ell^2(\mathbb{R}^N)}^2 + \frac{1}{2M} \|\mathbf{C}\boldsymbol{\theta} - \mathbf{u}_0\|_{\ell^2(\mathbb{R}^M)}^2, \quad (\text{soft constrained})$$

$$\text{Loss}_h(u(\mathbf{z}; \boldsymbol{\theta})) = \frac{1}{2N} \|\mathbf{A}_\phi \mathbf{A}\boldsymbol{\theta} + \mathbf{A}_{\phi_t} \mathbf{B}\boldsymbol{\theta} - \tilde{\mathbf{f}}\|_{\ell^2(\mathbb{R}^N)}^2, \quad (\text{hard constrained})$$

where

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

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

$$\mathbf{\Lambda}_\phi = \text{diag}(\phi(\mathbf{z}_1), \dots, \phi(\mathbf{z}_N)) \quad \text{and} \quad \mathbf{\Lambda}_{\phi_t} = \text{diag}(\phi_t(\mathbf{z}_1), \dots, \phi_t(\mathbf{z}_N)),$$

and the vectors

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

Therefore the Hessians of the losses are

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

Then the Hessians are bounded as

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

where $|\mathcal{L}_s|_2$ is the L^2 -operator norm of \mathcal{L}_s and

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

Proof. Once we have $u(\mathbf{y}; \boldsymbol{\theta}) = \mathbf{h}^\top(\mathbf{y})\boldsymbol{\theta}$, then the loss from the soft-constrained PINN can be re-written as

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

If we define the matrices

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

and the vectors

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

then $\text{Loss}_s(u(\boldsymbol{\theta})) = \frac{1}{2N} |\mathbf{A}\boldsymbol{\theta} - \mathbf{f}|_{\ell^2(\mathbb{R}^N)}^2 + \frac{1}{2M} |\mathbf{C}\boldsymbol{\theta} - \mathbf{u}_0|_{\ell^2(\mathbb{R}^M)}^2$. Similarly, for the loss

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

Since

$$\mathcal{L}_s[\psi + \phi u] = \phi(u_t + \mathcal{P}[u]) + \phi_t u + \psi_t, \quad \mathcal{P} \text{ is linear and } \phi(t, \mathbf{x}) = \phi(t).$$

Hence with

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

and the two diagonal matrices

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

we have $\text{Loss}_h(u(\boldsymbol{\theta})) = \frac{1}{2N} |\boldsymbol{\Lambda}_\phi \mathbf{A} \boldsymbol{\theta} + \boldsymbol{\Lambda}_{\phi_t} \mathbf{B} \boldsymbol{\theta} - \tilde{\mathbf{f}}|_{\ell^2(\mathbb{R}^N)}^2$. Then the two gradients are

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \text{Loss}_s(u(\boldsymbol{\theta})) &= \frac{1}{N} (\mathbf{A}^\top \mathbf{A} \boldsymbol{\theta} - \mathbf{A}^\top \mathbf{f}) + \frac{1}{M} (\mathbf{C}^\top \mathbf{C} \boldsymbol{\theta} - \mathbf{C}^\top \mathbf{u}_0), \\ \nabla_{\boldsymbol{\theta}} \text{Loss}_h(u(\boldsymbol{\theta})) &= \frac{1}{N} ((\boldsymbol{\Lambda}_\phi \mathbf{A} + \boldsymbol{\Lambda}_{\phi_t} \mathbf{B})^\top (\boldsymbol{\Lambda}_\phi \mathbf{A} + \boldsymbol{\Lambda}_{\phi_t} \mathbf{B}) \boldsymbol{\theta} - (\boldsymbol{\Lambda}_\phi \mathbf{A} + \boldsymbol{\Lambda}_{\phi_t} \mathbf{B})^\top \tilde{\mathbf{f}}).\end{aligned}$$

Differentiating w.r.t to $\boldsymbol{\theta}$ again, we obtain the Hessians as

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

Recall that $\mathcal{L}_s[u] = u_t + \mathcal{P}[u]$, hence by the property of an L^2 function/operator norm on \mathcal{L}_s , we have

$$|\mathcal{L}_s[u]|_2 \leq |\mathcal{L}_s|_2 |u|_2, \quad \text{This } |\cdot|_2 \text{ is a norm on functions.}$$

Then we have $|\mathbf{A}^\top \mathbf{A}|_2 \leq |\mathcal{L}_s|_2^2 |\mathbf{B}|_2^2$ (Here $|\mathbf{A}|_2$ is a matrix norm). And if we define

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

We can have

$$\begin{aligned}|\mathbf{A}^\top \boldsymbol{\Lambda}_\phi^2 \mathbf{A}|_2 &\leq C_1 |\mathbf{A}^\top \mathbf{A}|_2 \leq |\mathcal{L}_s|_2^2 |\mathbf{B}|_2^2, \\ |\mathbf{A}^\top \boldsymbol{\Lambda}_\phi \boldsymbol{\Lambda}_{\phi_t} \mathbf{A}|_2 &\leq C_2 |\mathbf{A}^\top \mathbf{B}| \leq C_2 |\mathcal{L}_s|_2 |\mathbf{B}|_2^2, \\ |\mathbf{B}^\top \boldsymbol{\Lambda}_{\phi_t} \mathbf{B}|_2 &\leq C_3 |\mathbf{B}^\top \mathbf{B}|_2 \leq C_3 |\mathbf{B}|_2^2\end{aligned}$$

Putting them back in, we obtain the final bounds

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

□

B Examples: Additional Details

We provide additional details for the ablation study, comparison study, 2D application and sensitivity study of m here. We also explain why each PDE or PDE system is chosen due to their particular difficulties for solvers in each sub-section.

B.1 Implementation Details

All experiments were implemented in PyTorch (*v2.2*) and executed on both CUDA-enabled NVIDIA GPUs and 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 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 functions. The same initial weights were used across all ablation experiments for consistency.

Model training was carried out using a two-stage strategy: first, we trained using the Adam optimizer with an initial learning rate of 10^{-3} for 50,000 steps. This was followed by fine-tuning with the L-BFGS-B optimizer using a learning rate of 1 for 5,000 steps. We did not perform hyperparameter sweeps, and the total number of training epochs was fixed across all experiments.

This paper does not rely on any 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 same sampling configuration was used consistently 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.

Ablation studies were conducted by disabling one component of our method at a time, while keeping all other settings fixed, to evaluate its contribution using relative L^2 and relative L^1 error metrics, which measure the discrepancy between the predicted and reference solutions in average (energy-based) and absolute (mass-based) senses, respectively. The relative form normalizes the error with respect to the reference solution, enabling fair comparisons across different scales and variables.

We also compare HC-PINN with causal training (MLP) and enhanced RBA using the same data and a total of 50k iterations, instead of the 300k iterations used in the original works. For the sensitivity study, experiments with different m values are conducted on the same points, using the same number of iterations and the same 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 is the same as in the 1D training.

B.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 [Bazant \(2017\)](#). It has numerous practical applications across a range of fields, including materials science [Allen & Cahn \(1979\)](#); [Shen & Yang \(2010\)](#), biological systems [Hyman et al. \(2014\)](#); [Takatori & Brady \(2015\)](#), and electrochemical systems [Horstmann et al. \(2013\)](#); [Tian et al. \(2015\)](#). We begin with the 1-dim Allen-Cahn equation with periodic boundary conditions formulated as follows:

$$\begin{aligned} u_t - \gamma_1 u_{xx} + \gamma_2(u^3 - u) &= 0, \quad (t, x) \in (0, T] \times (a, b), \\ u(0, x) = u_0(x), \quad x \in [a, b], \quad u(t, a) = u(t, b), \quad t \in [0, T], \end{aligned} \tag{5}$$

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.

B.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 [Raissi et al. \(2019\)](#). As shown in Table 2, 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 2. 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 4 presents cross-sectional comparisons at different times, further confirming the excellent agreement between the predicted and true solutions across the entire time horizon. These results provide strong evidence for the effectiveness of the proposed training strategy and underscore the critical role of each pipeline component.

B.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 2). However, once the initial condition enforcement and periodic boundary neural networks are incorporated,

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 2: *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.

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} . 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.

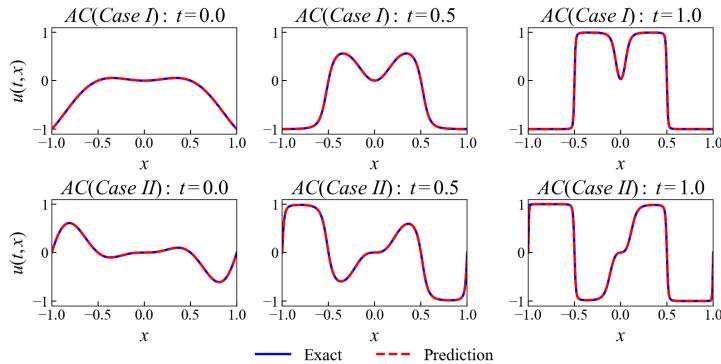


Figure 4: Best solution: *Allen-Cahn*.

B.3 Kuramoto-Sivashinsky PDE

Kuramoto–Sivashinsky (KS) equation is a fourth-order nonlinear partial differential equation, known for its chaotic behavior Kudryashov (1990); Michelson (1986). 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:

$$\begin{aligned} u_t &= -u_{xx} - u_{xxxx} - uu_x, \quad (t, x) \in (0, T] \times (a, b), \\ u(0, x) &= \cos\left(\frac{x}{16}\right)(1 + \sin \frac{x-1}{16}), \quad x \in [a, b], \quad u(t, a) = u(t, b), \quad t \in [0, T], \end{aligned} \tag{6}$$

where $T = 20$, $a = 0$, $b = 32\pi$. The best results are shown in 5:

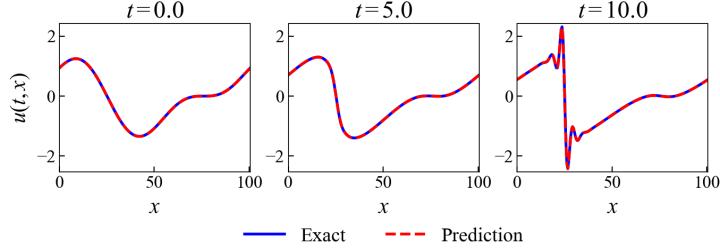


Figure 5: Best solution: *Kuramoto-Sivashinsky*.

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 3: *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.

As in Table 3, 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 contrast to the Allen-Cahn equation discussed in Section B.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.

B.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 Cahn & Hilliard (1958); Kim, Lee, Choi, Lee, Jeong, et al. (2016); Lee et al. (2014); Miranville (2017). The equation is expressed as:

$$\begin{aligned} u_t &= \epsilon_1(-u_{xx} - \epsilon_2 u_{xxxx} + (u^3)_{xx}), \quad (t, x) \in (0, T] \times (-L, L), \\ u(0, x) &= u_0(x), \quad x \in [-L, L], \\ u(t, -L) &= u(t, L), \quad t \in [0, T], \end{aligned} \tag{7}$$

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.

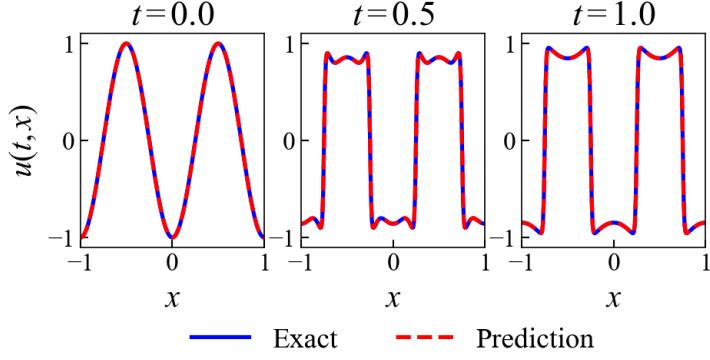


Figure 6: Best solution of the *Cahn-Hilliard*. The resulting relative L^2 error is 3.53×10^{-4} .

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 4. The best results are illustrated in Figure 6.

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 4: *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.

B.5 Gray-Scott Equation

Reaction and diffusion of chemical species can produce a variety of patterns (De Kepper et al., 1991; Nishiura & Ueyama, 1999), reminiscent of those often seen in nature. The Gray-Scott type system is one of classical mathematical models for chemical reactions (Gray and Scott, 1983,8; Liang, Jiang, Liu, Wang, and Zhang, 2022). The general irreversible Gray-Scott equations describe such reactions:



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

$$\begin{aligned} u_t &= \epsilon_1 u_{xx} + b(1 - u) - uv^2, \\ v_t &= \epsilon_2 v_{xx} - (b + k)v + uv^2, \quad (t, x) \in (0, T] \times (-L, L), \\ u(0, x) &= u_0(x), \quad v(0, x) = v_0(x), \quad x \in [-L, L], \\ u(t, -L) &= u(t, L), \quad v(t, -L) = v(t, L), \quad t \in [0, T], \end{aligned} \tag{9}$$

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:

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

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 5, 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 Figures 7 and 8, respectively.

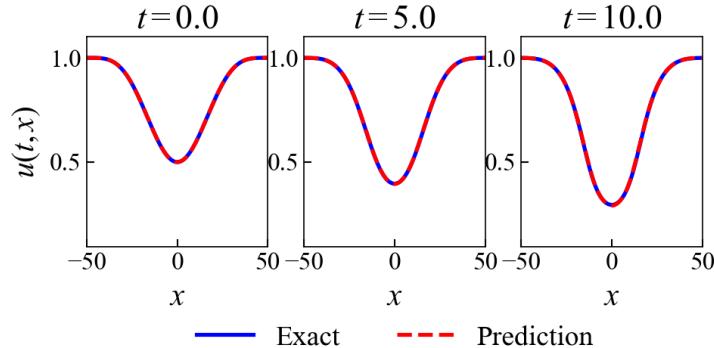


Figure 7: Best solution of the *Gray-Scott* (u species). The resulting relative L^2 error is 2.05×10^{-4} .

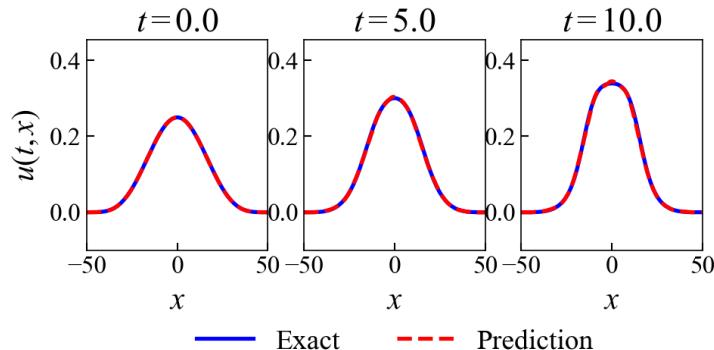


Figure 8: Best solution of the *Gray-Scott* (v species). The resulting relative L^2 error is 8.46×10^{-4} .

Ablation Settings					Performance	
PBC	IC1	IC2	SA	mini-batch	$\mathcal{E}_2 : u$	$\mathcal{E}_2 : v$
✓	✓	✗	✓	✓	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 5: *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.

B.6 Nonlinear Schröedinger Equation

In theoretical physics, nonlinear Schröedinger equation is a nonlinear PDE, applicable to classical and quantum mechanics (Fibich, 2015; Kato, 1987; Kevrekidis et al., 2001). The dimensionless equation of the classical field is

$$\begin{aligned} u_t &= iu_{xx} + i|u|^2u, \quad (t, x) \in [0, 2] \times [-\pi, \pi], \\ u(0, x) &= u_0(x), \quad x \in [-\pi, \pi], \\ u(t, -\pi) &= u(t, \pi), \quad t \in [0, 2], \end{aligned} \tag{10}$$

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:

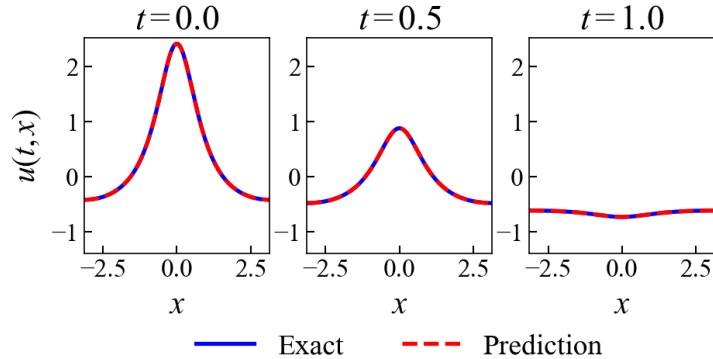


Figure 9: Best real-part solutions of the *Nonlinear Schröedinger Equation*.

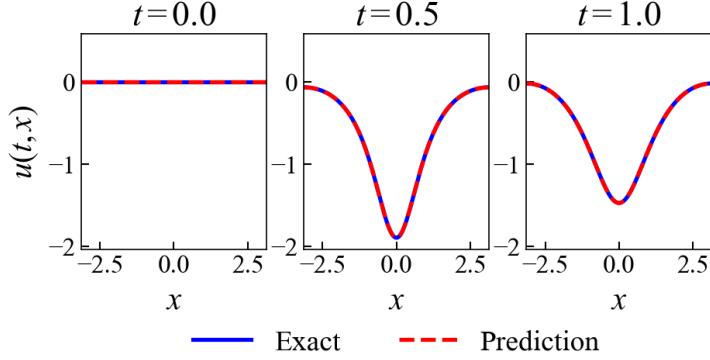


Figure 10: Best imaginary-part solutions of the *Nonlinear Schrödinger Equation*.

Table 6 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.

Ablation Settings					Performance: \mathcal{E}_2	
PBC	IC1	IC2	SA	mini-batch	Real part	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 6: *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.

B.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 [LeVeque \(2002\)](#); [Toro \(2013\)](#). It describes the propagation of a scalar profile without deformation and is given by:

$$\begin{aligned}
 u_t + cu_x &= 0, & (t, x) \in (0, T] \times (0, 2\pi), \\
 u(0, x) &= \sin(x), & x \in [0, 2\pi], \\
 u(t, 0) &= u(t, 2\pi), & t \in [0, T].
 \end{aligned} \tag{11}$$

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 3. *Let $u(t, x)$ denote the solution to the linear advection equation (11). With the transformed neural approximation in (3) and ψ and η satisfying the conditions in (4), then $u(t, x)$ satisfies the PDE iff η satisfies the forced linear transport equation:*

$$\eta_t + c\eta_x = -(\psi_t + c\psi_x).$$

This PDE is well-posed and stable for PINN training provided the following conditions hold:

- The function $f(t, x) := -(\psi_t + c\psi_x)$ satisfies $f \in C^1([0, T] \times \Omega)$.
- If $f(t, x)$ is uniformly bounded or decays in time, then the solution η remains stable and avoids growth-induced overfitting.

For our choice, we have $\psi(t, x) = u_0(x)e^{-\alpha t}$ and $\phi(t, x) = t$ with $\alpha > 0$, ensuring $\eta(0, x) = 0$. Then, the forcing term satisfies $\eta_t + c\eta_x = (\alpha \sin x - c \cos x)e^{-\alpha t}$. Since this forcing is smooth and decays exponentially in time, it guarantees both the stability and regularity of the neural network $\eta(t, x)$. 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.

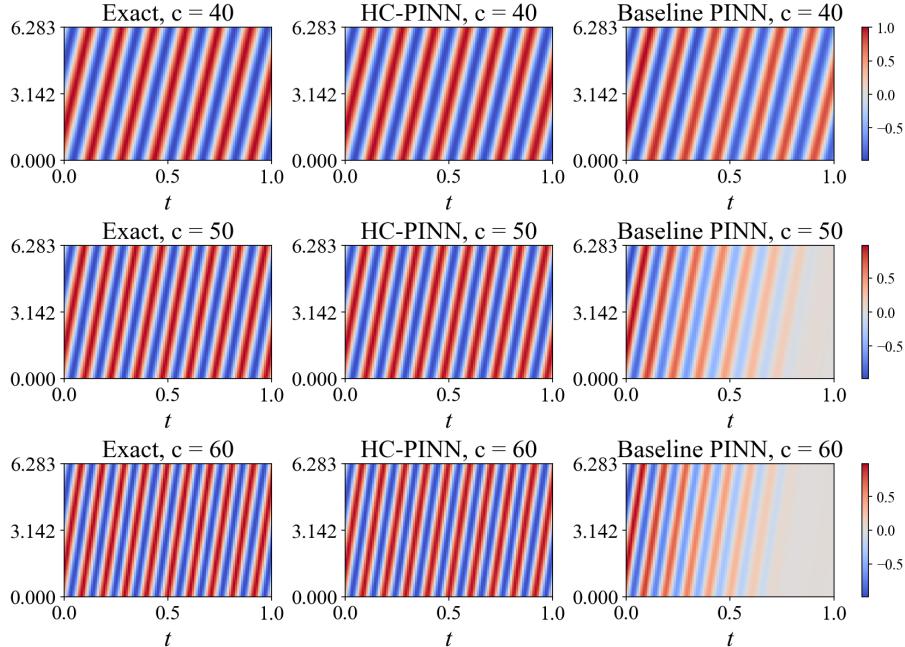


Figure 11: Periodic advection results after 50,000 iterations.

The implementation details are the same as in Section B.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 7 shows the averaged relative L_1/L_2 -error over 10 independent repetitions. As the advection velocity increases and the PDE becomes stiffer, making it harder for baseline PINN to train. However, our method consistently produces better results. This trend is clearly reflected in Table 7. 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 base line PINN fails to approximate the solution, while our method still produces stable and good results. This behavior is evident in Table 7 and Figure 11. Table 8 shows the ablation study of linear advection equation.

Solver	c	Rel. L_1 error	Rel. L_2 error
HC-PINN	30	0.0030	0.0041
	40	0.0141	0.0159
	50	0.0147	0.0164
	60	0.0171	0.0182
CINN(Braga-Neto, 2022)	30	0.0478	0.0579
	40	0.0714	0.0852
	50	0.4692	0.5365
	60	0.7293	0.7134
Baseline	30	0.0873	0.1003
	40	0.3918	0.4395
	50	0.7140	0.7797
	60	0.8339	0.8664

Table 7: 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}	1.03×10^{-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 8: *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.