

Causally enhanced initial conditions: A novel soft constraints strategy for physics informed neural networks

Wenshu Zha(查文舒), Dongsheng Chen(陈东升)†, Daolun Li(李道伦), Luhang Shen(沈路航), and Enyuan Chen(陈恩源)

Citation: Chin. Phys. B, 2025, 34 (**4**): 040701. DOI: 10.1088/1674-1056/adacd0
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Deep learning-assisted common temperature measurement based on visible light imaging

Jia-Yi Zhu(朱佳仪), Zhi-Min He(何志民), Cheng Huang(黄成), Jun Zeng(曾峻), Hui-Chuan Lin(林惠川), Fu-Chang Chen(陈福昌), Chao-Qun Yu(余超群), Yan Li(李燕), Yong-Tao Zhang(张永涛), Huan-Ting Chen(陈焕庭), and Ji-Xiong Pu(蒲继雄)

Chin. Phys. B, 2024, 33 (**8**): 080701. DOI: 10.1088/1674-1056/ad4cd9

Physical information-enhanced graph neural network for predicting phase separation

Yaqiang Zhang(张亚强), Xuwen Wang(王煦文), Yanan Wang(王雅楠), and Wen Zheng(郑文)

Chin. Phys. B, 2024, 33 (**7**): 070702. DOI: 10.1088/1674-1056/ad4328

TCAS-PINN: Physics-informed neural networks with a novel temporal causality-based adaptive sampling method

Jia Guo(郭嘉), Haifeng Wang(王海峰), Shilin Gu(古仕林), and Chenping Hou(侯臣平)

Chin. Phys. B, 2024, 33 (**5**): 050701. DOI: 10.1088/1674-1056/ad21f3

Advances in neuromorphic computing: Expanding horizons for AI development through novel artificial neurons and in-sensor computing

Yubo Yang(杨玉波), Jizhe Zhao(赵吉哲), Yinjie Liu(刘胤洁), Xiayang Hua(华夏扬), Tianrui Wang(王天睿), Jiyuan Zheng(郑纪元), Zhibiao Hao(郝智彪), Bing Xiong(熊兵), Changzheng Sun(孙长征), Yanjun Han(韩彦军), Jian Wang(王健), Hongtao Li(李洪涛), Lai Wang(汪莱), and Yi Luo(罗毅)

Chin. Phys. B, 2024, 33 (**3**): 030702. DOI: 10.1088/1674-1056/ad1c58

Synthesization of high-capacity auto-associative memories using complex-valued neural networks

Yu-Jiao Huang(黄玉娇), Xiao-Yan Wang(汪晓妍), Hai-Xia Long(龙海霞), Xu-Hua Yang(杨旭华)

Chin. Phys. B, 2016, 25 (**12**): 120701. DOI: 10.1088/1674-1056/25/12/120701

Causally enhanced initial conditions: A novel soft constraints strategy for physics informed neural networks

Wenshu Zha(查文舒), Dongsheng Chen(陈东升)†, Daolun Li(李道伦),
Luhang Shen(沈路航), and Enyuan Chen(陈恩源)

School of Mathematic, Hefei University of Technology, Hefei 230009, *China*

(Received 18 December 2024; revised manuscript received 10 January 2025; accepted manuscript online 22 January 2025)

Physics informed neural networks (PINNs) are a deep learning approach designed to solve partial differential equations (PDEs). Accurately learning the initial conditions is crucial when employing PINNs to solve PDEs. However, simply adjusting weights and imposing hard constraints may not always lead to better learning of the initial conditions; sometimes it even makes it difficult for the neural networks to converge. To enhance the accuracy of PINNs in learning the initial conditions, this paper proposes a novel strategy named causally enhanced initial conditions (CEICs). This strategy works by embedding a new loss in the loss function: the loss is constructed by the derivative of the initial condition and the derivative of the neural network at the initial condition. Furthermore, to respect the causality in learning the derivative, a novel causality coefficient is introduced for the training when selecting multiple derivatives. Additionally, because CEICs can provide more accurate pseudo-labels in the first subdomain, they are compatible with the temporal-marching strategy. Experimental results demonstrate that CEICs outperform hard constraints and improve the overall accuracy of pre-training PINNs. For the 1D-Korteweg–de Vries, reaction and convection equations, the CEIC method proposed in this paper reduces the relative error by at least 60% compared to the previous methods.

Keywords: initial condition, physics informed neural networks, temporal march, causality coefficient

PACS: 07.05.Mh, 02.30.Jr, 84.35.+i

DOI: 10.1088/1674-1056/adacd0 **CSTR:** 32038.14.CPB.adacd0

1. Introduction

Partial differential equations (PDEs) are ubiquitous in natural sciences, as they describe various phenomena that depend on rates of change in space and time. For example, convection equations, heat conduction equations and diffusion equations are PDE models for communication systems, acoustic behavior, fluid dynamics, heat dissipation, weather patterns, disease progression, and cellular dynamics. The time evolution of a typical PDE can be viewed as a sequence of processes that act on linear and nonlinear combinations of spatial differential operators. The numerical solution of PDEs is a challenging and active area of research in applied mathematics and computational science. PDEs also play a crucial role in modeling complex phenomena.

Neural networks constitute a highly popular field at present, demonstrating exceptional performance across various domains, such as computer vision, natural language processing, and the petroleum industry. For example, Zha[27,28] applied neural networks to the petroleum sector and achieved remarkable results. Further, neural networks have been increasingly employed in recent years for the numerical solution of PDEs, owing to their robust function approximation capabilities. Various methodologies have been proposed to utilize neural networks for approximating PDE solutions or inferring parameters within PDE models. For instance, Lagaris[1] introduced artificial neural networks (ANNs) to approximate the solutions of PDEs by using spatial coordinates as inputs. Dong[2] applied convolutional neural networks (CNNs) to learn the differential operators associated with PDEs and facilitate parameter inversion. Liu[3] developed a comprehensive framework for addressing initial value and boundary value problems in differential equations through automatic differentiation techniques. Sirignano[4] presented the deep Galerkin method, demonstrating that neural networks can achieve high-accuracy solutions for a specific class of quasi-linear parabolic PDEs. DeepOnet[5] and FNO[6] learned a family of PDEs using neural networks through operator learning.

The emergence of deep learning frameworks, such as PyTorch and TensorFlow, has facilitated the utilization of automatic differentiation tools that can efficiently and accurately compute the partial derivatives of variables at any given point. Raissi[7] leveraged this capability to incorporate the partial differential loss of an equation into the loss function of a neural network, coining the term physics informed neural networks (PINNs).

PINNs can be generalized to handle various types of differential equations. Pang[8] extended PINNs to fractional PINN (fPINNs) for solving fractional differential equations (FDEs). Shen[9] successfully used PINNs to solve the single-phase seepage equation. Mao[10] used PINNs with

†Corresponding author. E-mail: 2022111481@mail.hfut.edu.cn

© 2025 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

residual/gradient-based adaptive sampling methods to solve PDEs with sharp solutions. In addition, Wang,[26] Liu,[29] Tian[30] and Peng[31] have all enhanced the performance of PINNs through diverse methodologies, thereby rendering PINNs more versatile and broadly applicable.

In PINNs, the residual of PDEs typically receives most attention. Numerous experts have sought to enhance the performance of PINNs by improving the distribution of PDE residuals, as seen in approaches such as causally-PINNs[11] and failure-informed PINNs.[12] Residual adaptive refinement[13] and residual adaptive distribution[14] guide the distribution of PINN collection points by equation residuals. Additionally, some researchers have imposed stronger regularity conditions on PDE residuals, exemplified by gradient-enhanced PINNs.[15]

However, there is a notable scarcity of research focused on enhancing the learning process for the initial conditions. This can be attributed to two primary reasons: first, utilizing surrogate neural networks to impose strict constraints on the initial conditions is often regarded as a generally feasible method; second, an overemphasis on learning initial conditions may inadvertently lead to insufficient learning regarding other loss terms.

However, through experiments in this paper, some possible failure scenarios are found when using surrogate models for hard constraints on initial conditions. In addition, Wang[16] analyzed the convergence rate of PINNs using the neural tangent kernel (NTK).[17] Because the eigenvalue of the initial condition loss is lower, the importance of initial condition learning is emphasized.

Furthermore, with time, temporal marching strategies have been increasingly proposed, such as BC-PINNs,[18] time-march,[19] seq2seq,[20] PT-PINNs[21] and X-PINN.[22] A common feature of such temporal marching strategies is to start training in a subdomain near the initial conditions, generate pseudo-labels at the end of training, and then proceed with training over the entire domain of definition. This feature makes the learning of the initial conditions more important due to error accumulation.

In this context, to enhance the accuracy of PINNs in learning the initial conditions, this paper introduces a novel strategy named causally enhanced initial conditions (CEICs). This strategy imposes an additional loss on PINNs, which is constructed based on the derivative of the initial condition and the derivative of the neural network output at the initial condition. Experiments demonstrate that this approach exhibits superior ability in learning initial conditions compared to adjusting weights and applying hard constraints.

In particular, the incorporation of higher-order derivatives allows for stronger constraints to be imposed on the initial conditions. To address the training challenges associated with multiple derivative constraints, we introduce a causality coefficient as a hyperparameter when using multiple derivative terms for training. Furthermore, experiments show that CEICs can be effectively combined with the temporal marching strategy. In this paper, pre-training (PT)-PINNs combined with CEICs are used as an illustrative example.

This paper is organized as follows. Section 2 introduces PINNs and the temporal marching strategy, analyzes the importance of the initial conditions using NTK theory, and experimentally verifies challenges with hard constraints on the initial conditions using the convection equation. Section 3 demonstrates the use of CEICs to regularize the initial conditions in PINNs, discussing the use of causal coefficients in multi-derivative regularizations. Section 4 compares CEICs and their combination with PT-PINNs in terms of convection, reaction, and Korteweg–de Vries equation (KdV) equations, showing that CEICs have stronger universality and effective integration with the temporal marching strategy.

2. Preliminary

2.1. PINNs

Let us briefly review PINNs. The time evolution of PDEs can be uniformly expressed in the following form:

$$\begin{aligned} &\mathcal{A}[x,t;u(x,t)]=0,\ (x,t)\in\Omega,\\ &\mathcal{B}[x,t;u(x,t)]=0,\ (x,t)\in\partial\Omega,\\ &\mathcal{I}[x,t;u(x,t)]=0,\ t=0, \end{aligned} \tag{1}$$

where $u(\boldsymbol{x})$ is the unknown PDE solution, $\mathcal{A}$ is operator in domain of function $u(\boldsymbol{x})$, $\mathcal{B}$ and $\mathcal{I}$ are boundary and initial operator, respectively.

The PINN is a neural network framework proposed by Raissi.[7] It leverages the automatic differentiation function of high-performance deep learning frameworks such as TensorFlow and PyTorch. The fundamental concept involves employing a fully connected neural network to compute the partial derivative of the neural network $N(\boldsymbol{x},\theta)$ with respect to $\boldsymbol{x}$ through automatic differentiation. This partial derivative is then substituted into the PDE to evaluate the loss, which subsequently updates the network parameter θ.

There are three loss terms in the training PINNs: equation loss, initial loss and boundary condition loss, sampling $\{c_i\}_{i=1,2\ldots M_c}$ in Ω, $\{b_i\}_{i=1,2\ldots M_b}$ in $\partial\Omega$, $\{x_i\}_{i=1,2\ldots M_x}$ at $t=0$, the losses can be written as follows:

$$\begin{aligned} &\mathcal{L}_{\mathrm{e}}=\frac{1}{M_c}\sum_{i=1}^{M_c}|\mathcal{A}[c_i;u(c_i)]|^2,\ \ \mathcal{L}_{\mathrm{b}}=\frac{1}{M_b}\sum_{i=1}^{M_b}|\mathcal{B}[b_i;u(b_i)]|^2,\\ &\mathcal{L}_{\mathrm{i}}=\frac{1}{M_x}\sum_{i=1}^{M_x}|\mathcal{I}[x_i;u(x_i)]|^2,\ \ \mathcal{L}=\omega_{\mathrm{i}}\mathcal{L}_{\mathrm{i}}+\omega_{\mathrm{e}}\mathcal{L}_{\mathrm{e}}+\omega_{\mathrm{b}}\mathcal{L}_{\mathrm{b}}. \end{aligned} \tag{2}$$

Here M_c, M_b and M_x represent the sampling numbers of collection point, ω represents the weight corresponding to each loss term, finally, $N(\boldsymbol{x},\theta)$ will be updated by the backpropagation of L.

2.2. The importance of the initial conditions

The loss function for training PINNs typically combines three components: the initial condition loss, the equation loss, and the boundary loss. However, the relative influence of these components on the performance of PINNs during training across the entire domain is not fully understood.

Wang[11] analyzed the training process of PINNs using the NTK and found that the eigenvalues of the PDE residuals tend to be significantly larger than the eigenvalues of the initial conditions. This results in a faster convergence rate for the PDE residuals compared to the rate of fitting the initial conditions. In some cases, the eigenvalues of the initial conditions can even dominate the eigenvalues of the PDE residuals.

Based on this analysis, it is suggested that the weights ω_i assigned to the higher loss components should be dynamically adjusted to ensure that the initial conditions converge faster. This could potentially improve the overall performance of PINNs by ensuring that all components of the loss function are adequately addressed during training.

Through the analysis of NTK theory, it is evident that enhancing the learning of initial conditions can significantly improve a neural network's ability to converge towards the correct solution.

On another note, in recent years, temporal decomposition strategies have gained widespread adoption for improving the performance of PINNs. Penwarden[24] has generalized this temporal decomposition strategy. The approach of progressive learning within the temporal domain is collectively referred to as a "temporal march strategy", which includes methods such as BC-PINN[19] and seq2seq-PINNs.[20]

Due to the temporal causality inherent in PDEs, the initial conditions exert a considerable influence on subsequent function values. By focusing on a narrower domain in time t, the temporal march strategy underscores the significance of accurately capturing the initial conditions.

In the temporal march strategy, training consistently begins in a subdomain close to the initial condition, as illustrated by network 1 in Fig. 1. The inherent temporal causality present in PDEs means that the initial conditions have a significant influence on subsequent function values. Within this subdomain, the impact of the initial conditions on the performance of network 1 is particularly critical. If network 1 is not trained effectively, its inaccuracies will propagate backward due to the causal relationship inherent in the PDEs, adversely affecting the performance of subsequent networks, such as network 2. Therefore, accurate learning of the initial conditions in the subdomain is essential to ensure the overall performance of the PINN in solving PDEs.

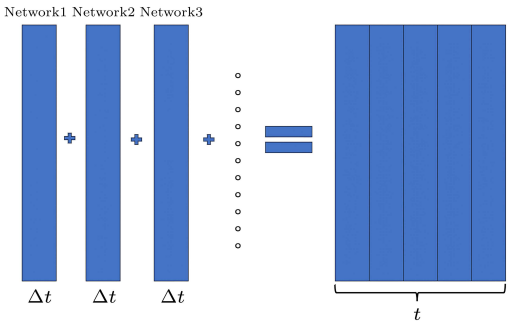


Fig. 1. Seq2seq2 learning strategy.

In summary, whether using the standard PINNs to approximate PDEs or using temporal march strategy to enhance the performance of PINNs, the initial condition learning is always very important.

2.3. The challenges with surrogate network

Based on the introduction in the previous section, it is clearly established that the accurate learning of the initial condition is of utmost importance. As such, it is crucial to develop a strategy to enhance the learning of this initial condition. One commonly adopted method to achieve this goal is the utilization of a surrogate network architecture within PINNs, which effectively enforces the initial condition. Specifically, the surrogate network outlined in Eq. (3) is widely employed, ensuring that the neural network unequivocally fulfills the initial condition at the initial time.

$$\text{output} = N(x,t,\theta)\times(1-\mathrm{e}^{t})+h(x,0). \tag{3}$$

However, when employing PINNs to approximate PDEs that undergo significant changes near the initial condition, a challenge arises concerning the surrogate networks' need to exhibit even more drastic changes compared to the original network. To illustrate this point, consider a 1D ordinary differential equation described by

$$u(0)=0,\ \ u'(x)=\frac{1}{10\sqrt[10]{x^9}}. \tag{4}$$

Obviously, $u(x)=\sqrt[10]{x}$, if used by the surrogate networks to approximate $u(0.01)\approx 0.63$, needs the network output value $\frac{0.63}{\mathrm{e}^{0.01}-1}-u(0)\approx 66.3$; in this case, the surrogate networks need to adopt more drastic changes, which makes it harder for the neural network to learn.

However, the challenges brought by surrogate networks are not limited to this. This can be observed when $(1-\mathrm{e}^{t})$ approaches 1, at which point this network output can be equivalent to the formula

$$\text{output} = N(x,t,\theta)+h(x,0). \tag{5}$$

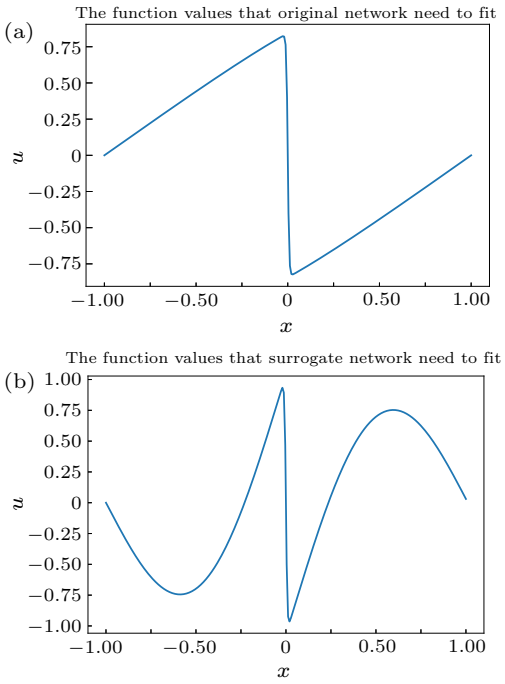


Fig. 2. Different functions need to be fitted when using different networks.

Assume that the PINNs want to approximate $f(x,t)$. This can be found from formula (5); the neural networks need to output $f(x,t)-h(x,0)$. This is a duality: on the one hand, it may make the learning of PINNs simpler; on the other hand, if the function values at subsequent time steps differ significantly from the initial conditions, this may make the learning of PINNs more challenging, as shown in Fig. 2. Figure 2(a) shows the function values in the Burgers equation at $t=0.8$, and it is obvious that learning Fig. 2(b) is harder than Fig. 2(a).

In the above, a potentially difficult learning scenario is presented when using a surrogate network. A specific example in PINNs is considered where the convection equation is learned using the surrogate; the convection equation is described as

$$\begin{aligned}&\frac{\partial u}{\partial t}+\beta\frac{\partial u}{\partial x}=0,\ \ x\in\Omega,\ \ t\in[0,0.1],\\&u(x,0)=h(x),\ \ x\in\Omega,\\&u(2\pi,t)=u(0,t).\end{aligned}\tag{6}$$

The $h(x)$ is $\sin(x)$ and β is 40 for this experiment, and Fig. 3 shows the solution in the corresponding time domain.

In this experiment, it was observed that utilizing the surrogate network to learn the convection equation led to inferior performance compared to only using the original network. This can be clearly seen in Fig. 4(a), where despite the surrogate network forcing the neural network to satisfy $h(x)$ during training, this method made it significantly more challenging to reduce the other two losses. Conversely, Fig. 4(b) depicts the loss reduction process during standard PINN training for the convection equation, where all losses were successfully minimized. The final loss values for both training methods are presented in Table 1.

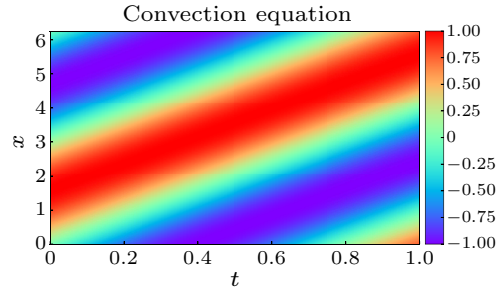


Fig. 3. The solution of convection equation at $t=[0,0.1]$.

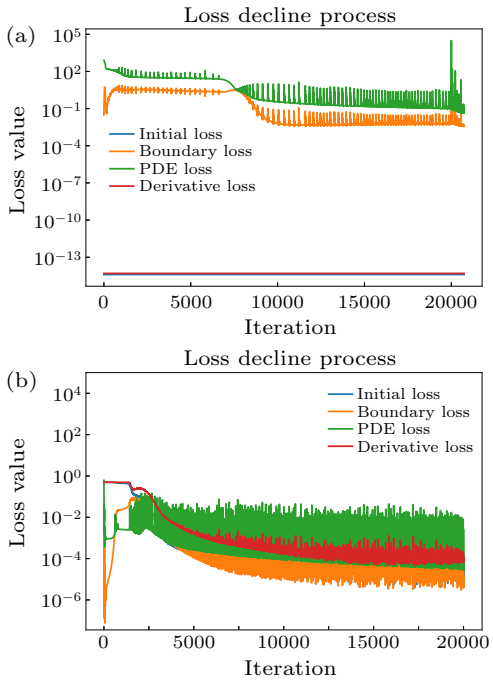


Fig. 4. Comparison of the loss decline process.

Table 1. Different loss terms in different cases.

	PINNs ($\omega_i=1$)	Surrogate network	Enhanced initial condition	PINNs ($\omega_i=10$)	PINNs ($\omega_i=100$)	PINNs ($\omega_i=1000$)	Adding collection point
PDE loss	3.57×10^{-5}	7.27×10^{-3}	1.29×10^{-5}	6.41×10^{-5}	1.99×10^{-4}	2.52×10^{-4}	4.08×10^{-5}
Boundary loss	1.18×10^{-5}	2.25×10^{-3}	2.95×10^{-6}	3.12×10^{-5}	5.32×10^{-5}	2.04×10^{-4}	3.57×10^{-5}
Initial loss	1.97×10^{-5}	0	6.43×10^{-8}	1.98×10^{-6}	2.34×10^{-6}	8.56×10^{-8}	1.06×10^{-5}
Derivative loss	3.62×10^{-4}	0	2.20×10^{-6}	4.38×10^{-5}	5.77×10^{-5}	1.18×10^{-6}	1.32×10^{-4}
Relative loss	7.25×10^{-3}	2.41×10^{-2}	6.77×10^{-4}	3.46×10^{-3}	6.78×10^{-3}	9.52×10^{-3}	5.74×10^{-3}
Time/1000 epoches	4.25 s	5.59 s	4.65 s	4.26 s	4.26 s	4.44 s	5.11 s

In summary, the utilization of a surrogate network can alter the output of the neural network, rendering it non-universal for certain PDEs. In some instances, this approach can even exacerbate the difficulty of approximating the PDEs. Consequently, there is a pressing need for a more generalized strategy to enhance the learning of the initial condition.

3. Causally enhanced initial conditions

3.1. Enhanced initial condition

This is inspired by G-PINN,[15] which indicates the derivatives of PDE residual also be zero. Afterwards, the gradient of PDE residuals is added to the loss function to enhance the learning of PDE residual.

Similarly, to enable PINNs to learn better in initial conditions, PINNs should not only satisfy the initial function $h(x)$ when $t = 0$, but also ensure that the derivative of the neural network at the initial time should be equal to $h'(x)$.

Overall, this paper introduces a novel loss function that incorporates the derivative of the initial condition and the derivative of the neural network output at the initial condition. By incorporating this additional loss term in the loss function, enhanced learning of the initial condition can be achieved.

This may lead to confusion, because the initial condition $h(x)$ is accurately fitted in standard PINNs, and consequently its corresponding $h'(x)$ should also be precisely approximated. However, since the learning of $h(x)$ in PINNs is discretized, the precision of $h'(x)$ in the discrete setting may, as in Fig. 5, not be satisfactory. In this case, regardless of how dynamically the weights are adjusted, overcoming this dilemma in learning initial conditions remains challenging.

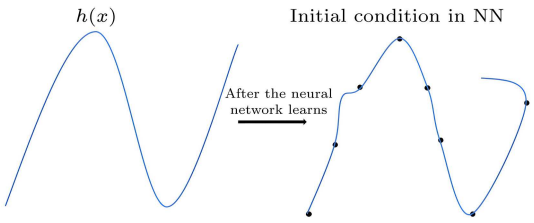


Fig. 5. The possible error when PINNs are learning the initial condition.

Our aim is to enhance the accuracy of learning by applying the high precision $h'(x)$ to influence the learning of $h(x)$, compared to the weight adjustment; this is another perspective of reinforcement learning for initial conditions.

The new loss in PINNs is described as

$$\mathcal{L} = \omega_{\mathrm{f}}\mathcal{L}_{\mathrm{f}} + \omega_{\mathrm{b}}\mathcal{L}_{\mathrm{b}} + \omega_{\mathrm{i}}\mathcal{L}_{\mathrm{i}} + \omega_{\mathrm{d}}\mathcal{L}_{\mathrm{d}}, \tag{7}$$

$$\mathcal{L}_{\mathrm{d}} = |N'(x,0) - h'(x)|^2. \tag{8}$$

In this paper, the proposed method is applied to convection equations and used to compare standard PINNs and PINNs with a surrogate network. In the experiment, all weights are set to 1 unless specified otherwise, and the network architecture consists of four hidden layers, each with 20 neurons. The optimizer is trained 20000 times using Adam and then fine-tuned with the L-BFGS (limited-memory Broyden–Fletcher–Goldfarb–Shanno optimizer). Table 1 presents a comparison of different loss terms and relative errors for seven different training strategies.

The enhanced initial condition (EIC) method achieves significantly lower initial and derivative losses, indicating a more precise and efficient performance. Figure 6 illustrates the loss reduction process when the PINNs employ the EIC method, and it can be observed that compared to Fig. 4, the loss is reduced more effectively.

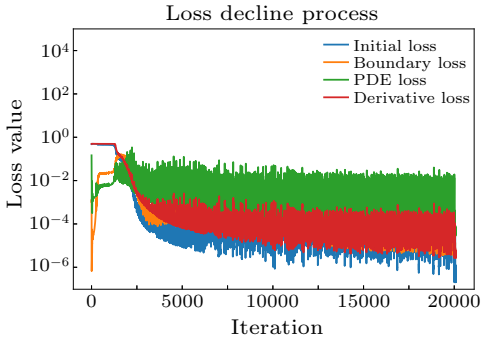


Fig. 6. Loss decline process when enhancing the initial condition.

It is also observed that when PINNs simply increase the weight of the initial condition, the relative loss and the initial condition loss are not as low as those achieved with the EIC method. This suggests that EIC reinforces the learning of the initial conditions in a different manner. Furthermore, by increasing the number of collection points at the initial time (from 256 to 1024), the initial condition loss (1.06×10^{-5}) is also not as good. Figure 7 compares the relative errors between the standard PINN and the PINN with an added collection initial enhancement (CIE).

This is also can found that when PINNs just increases the weight of the initial condition, the relative loss and the loss of initial condition are not better than those of the EIC, this indicates that the EIC reinforces learning of initial conditions in another way.

In addition, by adding the collection points at the initial time (from 256 to 1024), the initial condition loss (1.06×10^{-5}) is not as low as that obtained by the EIC, and the relative error (5.74×10^{-3}) is not as good. The relative error comparison between a PINN and a PINN with added CIE is shown in Fig. 7.

The results of this experiment indicate that the EIC is a viable alternative to the surrogate model. This strategy provides stronger regularization of the training of PINNs on initial conditions compared to adjusting weights or adding collection points.

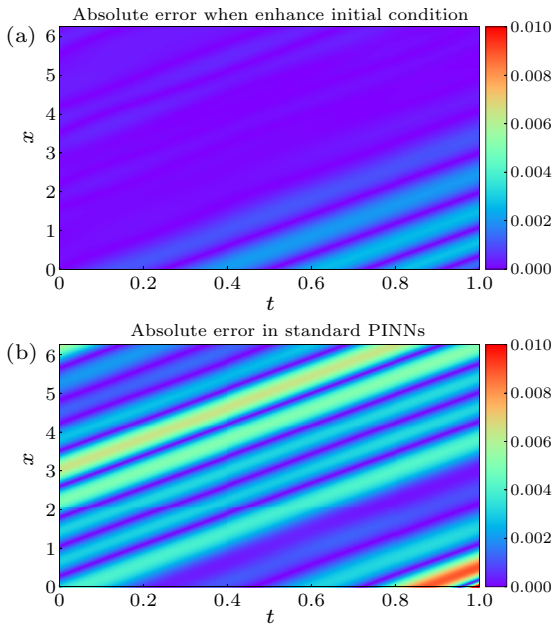


Fig. 7. The comparison of training result.

It is worth noting that since the calculation of the derivative is consistent with the calculation process of the PDE residual, the computational cost of adding derivative calculations for the sampling points in the initial condition is relatively low.

3.2. Enhancement of causality

The initial enhancements presented above demonstrate significant improvements in the performance of PINNs. Based on this strategy, a natural recursive idea emerges. If enhancing the initial condition by learning derivatives is effective, is it possible to further improve the initial condition by incorporating higher-order derivatives as loss terms? Specifically, one could consider adding the derivative of the derivative (or even higher-order derivatives) as additional loss terms to further regularize the training of PINNs.

As shown in Fig. 8, in contrast to indirect feedback regulation, we can directly exert stronger feedback on the initial conditions and their derivatives by incorporating additional derivative loss terms. This approach allows for more precise control over the initial conditions and their derivatives, potentially leading to improved performance of PINNs.

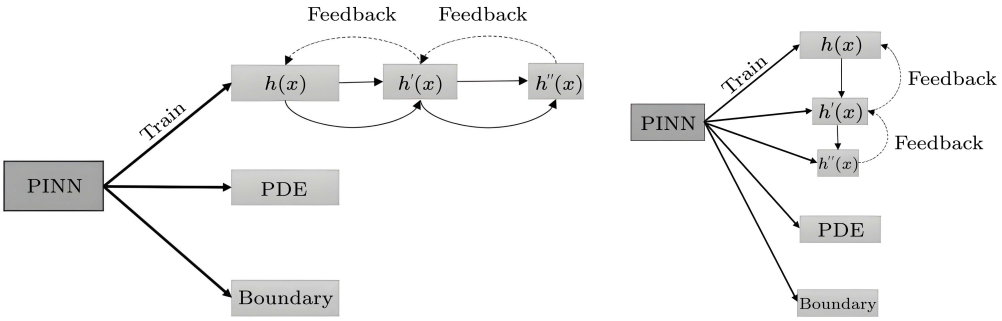


Fig. 8. Enhanced initial condition PINNs.

Table 2. Different loss terms when adds the second derivative.

	Initial loss	Boundary loss	PDE loss	Derivative loss	Second derivative loss
Multiple enhances	2.18×10^{-8}	1.76×10^{-6}	1.85×10^{-5}	6.65×10^{-7}	9.95×10^{-5}

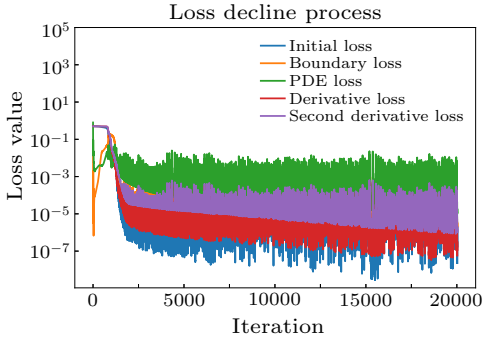


Fig. 9. Loss decline process when adding the second derivative.

This idea can be utilized to enhance the loss term mentioned above by incorporating a second derivative into the loss function. The experiments are presented in Fig. 9 and Table 2, demonstrating a relative error of 5.44×10^{-4}. However, it is important to note that the time cost increases from 4.65 seconds per 1000 epochs to 5.42 seconds per 1000 epochs. This increase is attributed to the fact that, in the original loss function, there was no necessity for automatic differentiation to compute the second derivative, which is quite time-consuming.

Adding more derivative terms to the loss function does not necessarily yield better results. Firstly, it can increase computational costs. Secondly, it may lead to suboptimal convergence due to the presence of multiple loss terms. Addressing the descent difficulties encountered by the loss function composed of multiple loss terms during the gradient descent process is a challenging problem. For example, when using enhanced initial conditions to train the one-dimensional Korteweg–de Vries (KdV) equation, as described by Eq. (9) and illustrated in Fig. 10, we observed that several loss terms were difficult to further reduce during the training process, as

shown in Fig. 11.

$$\begin{aligned} &u_t + uu_x + 0.0025u_{xxx} = 0, \ \ x \in (-1,1), \ \ t \in [0,0.3], \\ &u(x,0) = h(x) = \cos(\pi x), \ \ x \in (-1,1), \\ &u(1,t) = u(-1,t), \ \ t \in [0,0.3]. \end{aligned} \tag{9}$$

It can be observed from Fig. 11 that the PDE residual is notably challenging to reduce during the training process. An intriguing observation from the experiment is as follows: the loss associated with the second derivative (2.40×10^{-7}) is lower than that of the first derivative (2.91×10^{-7}), which clearly undermines the objective of learning the second derivative.

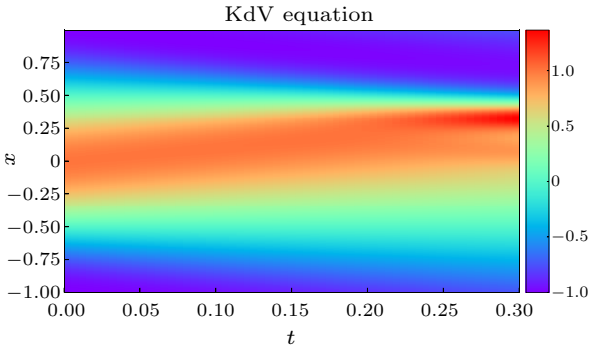


Fig. 10. The solution of KdV equation.

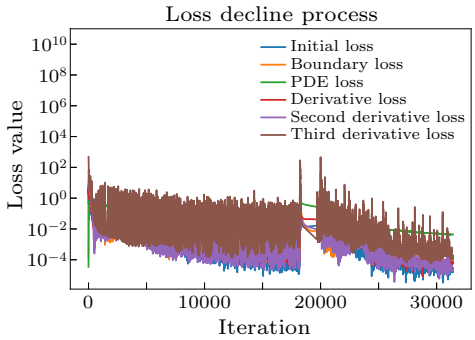


Fig. 11. Loss decline process when all hypermeters equal 1.

The training results' loss terms are presented in Table 3, revealing that, in this case, the inclusion of new losses does not yield as favorable learning outcomes as when no additional losses are introduced.

The objective of augmenting the regularization of PINNs by incorporating second-order derivatives is to improve the learning of initial conditions. However, neural networks tend to overly prioritize minimizing the loss associated with the second-order derivatives. As evidenced in Table 3, compared to standard PINNs, although the derivative values decrease, the loss of the initial conditions does not receive a more stringent constraint.

Perhaps choosing a suitable hyperparameter for each loss term would solve this training problem, but assigning a suitable hyperparameter to each of the six loss terms is obviously very difficult and time-consuming. The best idea is to let the neural networks learn the second derivative when the first derivative is learned sufficiently. On the other hand, if using the adaptive weight to learn multiple loss terms, it cannot respect the causal relationship between the initial condition and derivative term.

To respect this causality in the initial condition and derivative term, a causal coefficient is introduced to constrain the derivative term. This coefficient is shown in formula (11), and it can effectively ensure that only the first derivative is learned before learning the second derivative.

$$\mathcal{L} = \omega_{\mathrm{f}}\mathcal{L}_{\mathrm{f}} + \omega_{\mathrm{b}}\mathcal{L}_{\mathrm{b}} + \omega_{\mathrm{i}}\mathcal{L}_{\mathrm{i}} + \sum_{i=1}^{k} \omega_{\mathrm{d}_i}\mathcal{L}_{\mathrm{d}}^{i}, \tag{10}$$

$$\omega_{\mathrm{d}_i} = \frac{\mathrm{e}^{-\varepsilon^i f^{i-1}}}{\varepsilon^i}, \tag{11}$$

$$\mathcal{L}_{\mathrm{d}}^{i} = |N^{(i)}(x,0) - h^{(i)}(x)|^2. \tag{12}$$

Since the training process is different for different PDEs, the ε as a hyperparameter needs to be set before training and can be adapted to different PDEs. For the KdV equation, the ε is set at 10 in this paper.

Table 3. Comparison of different losses of non-causality and causality coefficient.

	Initial loss	Boundary loss	PDE loss	$h'(x)$ loss	$h''(x)$ loss	$h'''(x)$ loss	Relative error
Non-causality	7.58×10^{-8}	2.01×10^{-5}	9.30×10^{-4}	2.91×10^{-7}	2.40×10^{-7}	1.34×10^{-5}	1.10×10^{-2}
Causality	2.23×10^{-10}	9.66×10^{-8}	1.53×10^{-6}	1.20×10^{-8}	3.05×10^{-7}	6.10×10^{-5}	4.39×10^{-4}
Standard PINNs	1.90×10^{-8}	1.51×10^{-7}	2.18×10^{-6}	9.60×10^{-6}	8.25×10^{-3}	9.21×10^{-2}	3.62×10^{-3}

Figure 12 shows the loss-decline process when the causality enhances the initial condition to train PINNs for KdV equations in $t \in [0,0.3]$. From the descent process and Table 3, it can be seen that no loss value of any high-order derivative is higher than that of the low-order derivative; the PDE residual, boundary loss, and the initial loss can also drop to a lower order of magnitude.

Additionally, the ω_{d_i} will converge to ε^{-i} as the loss decreases, it is worth discussing how it differs from the causality coefficient if setting $\omega_{g_i} = \varepsilon^{-i}$ at the beginning. For this reason, we perform an experiment on the causality coefficient and the fixed coefficient 50 times to train a neural network for the KdV equation. The error bar is shown in Fig. 13. It is obvious from this experiment that using the causality coefficient is

more stable and achieves a lower mean error.

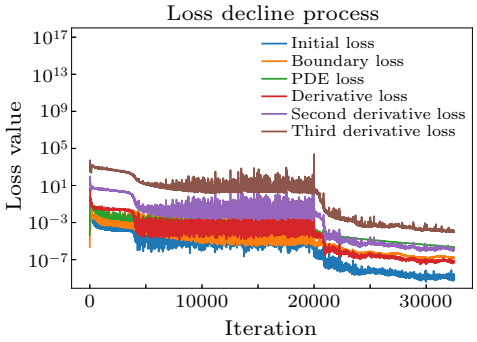


Fig. 12. Loss decline process when using causality coefficients.

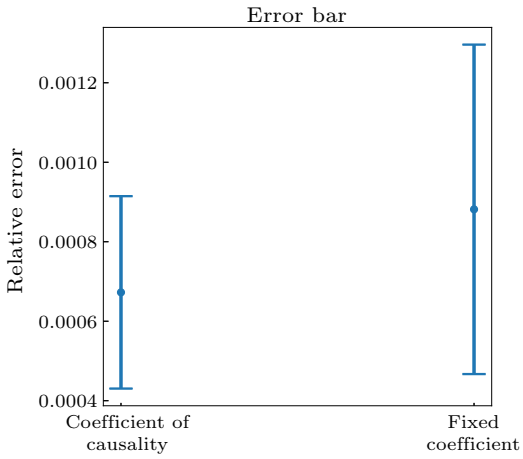


Fig. 13. Error bar for causality and the fixed coefficient.

4. Experiment

In the experiment, we used a fully connected neural network with four hidden layers and 20 neurons in each hidden layer. When choosing the optimizer, we first use the Adam optimizer with a learning rate of 0.01 to train 20000 epochs and then use the LBFGS optimizer with a learning rate of 10 to optimize again. The training set is composed of 2000 residual collection points using Latin hypercube sampling and 256 uniformly spaced boundary points for any PDE.

The total loss for any given model can be written as

$$\mathcal{L} = \omega_{\mathrm{f}}\mathcal{L}_{\mathrm{f}} + \omega_{\mathrm{b}}\mathcal{L}_{\mathrm{b}} + \omega_{\mathrm{i}}\mathcal{L}_{\mathrm{i}} + \omega_{\mathrm{d}_1}\mathcal{L}_{\mathrm{d}}^1 + \omega_{\mathrm{d}_2}\mathcal{L}_{\mathrm{d}}^2 + \omega_{\mathrm{d}_3}\mathcal{L}_{\mathrm{d}}^3. \quad (13)$$

The ω_{i} is set to 100 in this paper, ω_{f} and ω_{b} are set to 1.

Moreover, for the KdV and convection equations in the following experiment, we find that it is difficult to solve them when $t = [0,1]$ using PINNs directly, even with CEICs. In view of this, the temporal march strategy can be used to extend the good results in the previous experiment to the larger time domain. The temporal march strategy uses PT-PINNs, but only the pseudo-label generation strategy in PT-PINNs is used.

In the case of increasing the supervised data, our loss term will increase the supervised data, and the coefficients corresponding to the supervised data are set to 100. For more convenient reference, we refer to the causality enhancing initial as CEIC and the one using surrogate networks as hard control (HC).

When selecting a baseline to demonstrate the effectiveness of CEICs in enhancing the accuracy of temporal march strategy, we consider using standard PINNs, PT-PINNs, and enforcing hard constraints on both. In addition to that, we have set seq2seq-PINNs, causal-PINNs and ACSM-PINNs[25] as baselines.

4.1. KdV equation

In previous experiments, the KdV equation was successfully solved using CEICs. However, directly applying PINNs with CEICs to learn the KdV equation when $t \in [0,1]$ remained a challenge (Fig. 14). To overcome this issue, we employed a temporal marching strategy to extend the well-learned results from the earlier time domain to the later time domain. In this experiment, we utilized PT-PINNs to generalize the performance of PINNs across the entire time domain. For comparative purposes, we evaluated the error performance of three scenarios: adding HC, without any additional enhancements, and adding CEICs.

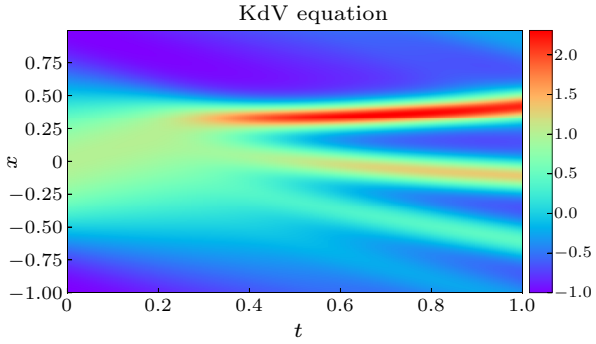


Fig. 14. The solution of the KdV equation at $t = [0,1]$.

Table 4. Error comparison in the KdV equation.

Strategy	Relative loss
Standard PINNs	6.21×10^{-1}
PINNs with CEICs	5.62×10^{-1}
PINNs with HC	5.89×10^{-1}
PT-PINNs with CEICs	5.70×10^{-3}
PT-PINNs with HC	5.20×10^{-2}
PT-PINNs	7.73×10^{-2}
Seq2seq-PINNs	1.57×10^{-2}
ACSM-PINNs	1.62×10^{-2}
Causal-PINNs	4.68×10^{-1}

When using PT-PINNs, we first trained the model in the $t = [0,0.3]$ interval and generated pseudo-labels. Subsequently, we trained the entire model in the [0,1] interval. Table 4 presents the different relative errors obtained using various strategies, and it can be observed that PT-PINNs with CEICs achieve higher accuracy compared to PINNs with the HC.

Figure 15 shows the different errors in different strategies. Panel (a) is PINNs with CEICs, while panel (b) is PINNs with the HC, both of which use a pre-training strategy.

For a more intuitive comparison, we have also provided three error accumulation results over time for the three different training strategies. As evident in Fig. 16, compared to PT-PINNs without the use of CEICs, the other two strategies incorporating initial conditions have achieved better learning performance. Furthermore, the CEIC strategy proposed in this paper exhibits superior performance in terms of error accumulation compared to the HC.

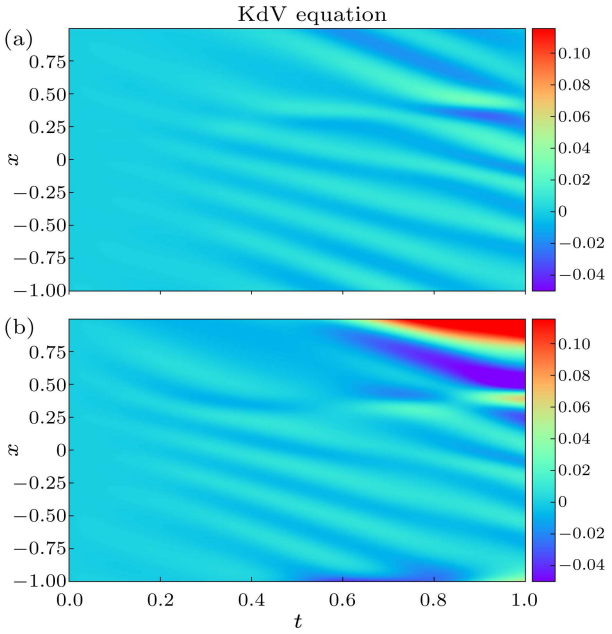


Fig. 15. Error comparison in different strategies.

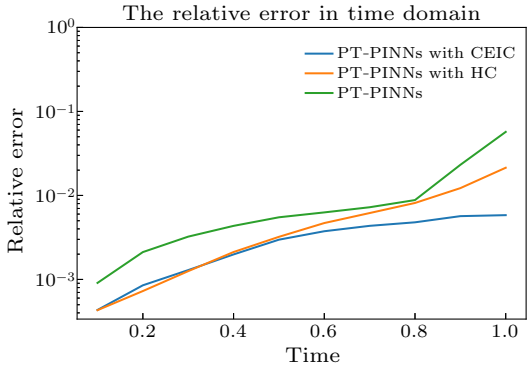


Fig. 16. The relative error in time domain.

Upon further analysis, while the performance near the initial condition is very similar for CEICs and the HC, differences emerge in the later time domain. As previously mentioned in Subsection 2.2, we speculate that this may be because, at a later time, the surrogate network needs to fit more complex values.

For an intuitive comparison, Fig. 17 shows the comparison between the function values that the neural network needs to fit, and the network output values in the two strategies. It can be observed that the function requiring fitting by the surrogate networks is more complex compared to the function requiring fitting by the original networks. This complexity may contribute to the lower accuracy of the HC compared to CEICs.

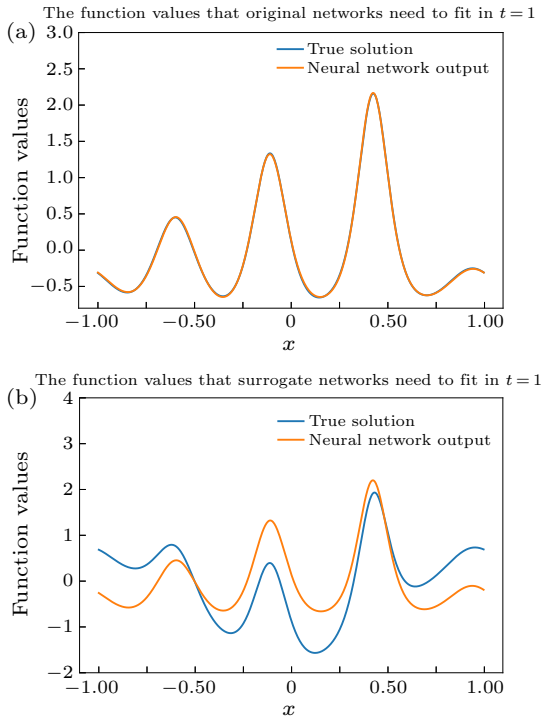


Fig. 17. Different functions needed to be fitted.

4.2. Reaction equation

The reaction equation has the following form:

$$\begin{aligned} &\frac{\partial u}{\partial t} - \rho u(1-u) = 0, \quad x \in \Omega, \quad t \in [0,T], \\ &u(x,0) = h(x), \ x \in \Omega. \end{aligned} \tag{14}$$

Here ρ is the reaction coefficient, and is set to $10, 20, 40$ in this paper, and $h(x)$ is the function expression of the initial time. Krishnapriyan[3] proposed that when $\rho > 5$, the reaction will be very difficult to learn using standard PINNs.

In this paper, we assume that $h(x)$ is a Gaussian distribution function $\exp[-\frac{(x-\pi)^2}{2(\frac{\pi}{4})^2}]$, and the corresponding boundary conditions as $u(0,t) = u(2\pi,t)$.

The different solution images corresponding to different ρ are shown in Fig. 18. Figures 18(a)–18(c) correspond to $\rho = 10, 20, 40$. With the increase of ρ, the changing trend of the function near the initial time domain becomes more drastic. This drastic trend may cause difficulties when training the neural network after adding hard constraints.

Table 5 presents the relative error of different strategies for various values of ρ. In addition to the relative error of PINNs with added causality enhancing the initial cases, it can be observed it is very challenging to directly solve the reaction equation using hard constraints.

Table 5. Error comparison in reaction equation.

	Standard PINNs	PINNs with CEIC	PINNs with HC	PT-PINNs with CEIC	PT-PINNs with HC	PT-PINNs	Seq2seq PINNs
$\rho = 10$	5.57×10^{-2}	3.11×10^{-2}	5.17×10^{-1}	1.51×10^{-2}	1.63×10^{-2}	2.12×10^{-2}	7.47×10^{-2}
$\rho = 20$	9.77×10^{-1}	2.21×10^{-2}	5.61×10^{-1}	3.13×10^{-3}	4.72×10^{-1}	1.20×10^{-2}	5.28×10^{-2}
$\rho = 40$	8.68×10^{-1}	2.25×10^{-2}	7.71×10^{-1}	2.18×10^{-3}	4.53×10^{-1}	6.32×10^{-3}	3.64×10^{-2}

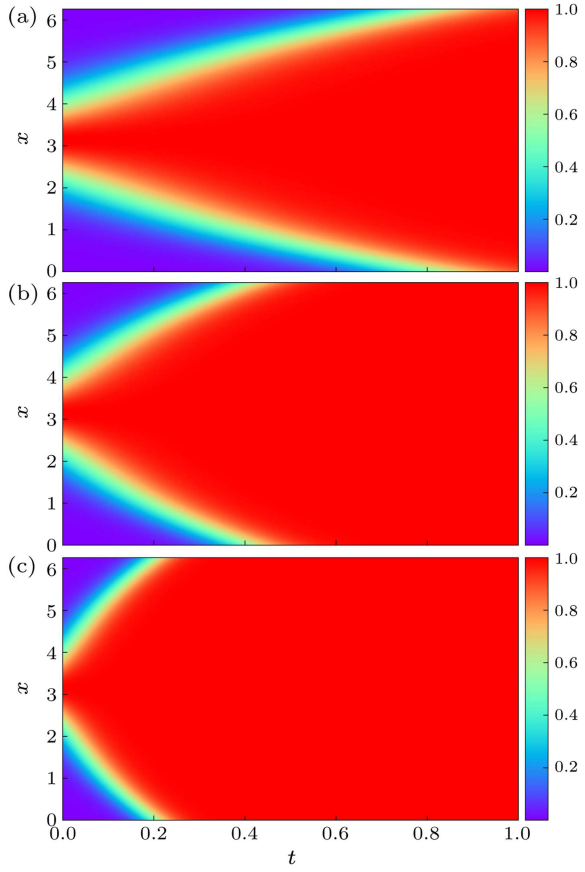


Fig. 18. The different solutions of different ρ.

Although directly using PINNs with CEICs roughly solves the reaction equation with a relative error of 2%, we find that the accuracy can be further improved by employing a pre-training strategy. Through experiments, we discover that in the case of low ρ values, the use of hard constraints combined with the pre-training strategy can address the issue of hard constraints not being applicable. However, overall, the CEIC strategy remains more versatile, even when ρ is higher.

For PT-PINN training on the reaction equation, we first train the neural network on $t = [0, 0.1]$ and obtain 256 pseudo-labels as supervision data, then train the entire time domain. Figure 19 shows the relative error of PT-PINNs in the time domain when using three different constraint methods. The hard constraint performs poorly compared to the other strategies during the pre-training phase and fails completely in the entire time domain. This aligns with the phenomenon that the surrogate network finds it more challenging to adapt to regions of dramatic change.

Furthermore, the CEIC strategy can obtain more accurate pseudo-labels by enhancing the learning of the initial conditions in the pre-training stage, thereby reducing the final relative error to 0.2%.

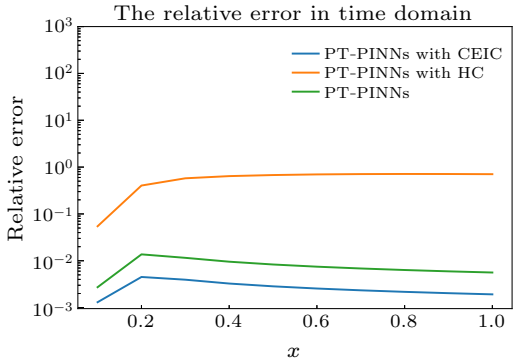


Fig. 19. The relative error in time domain.

4.3. Convection equations

The convection equation is a high frequency equation, and it has the characteristics of strong transformation in the time domain. In our previous experiments, we successfully enhanced the performance of PINNs in learning the convection equation by using CEICs. However, just like solving the KdV equation, we just solved it in a small-time domain. Based on this, we still use PT-PINNs with CEICs to solve the convection equation when $t \in [0, 1]$. The prediction results and error plots are presented in Fig. 20.

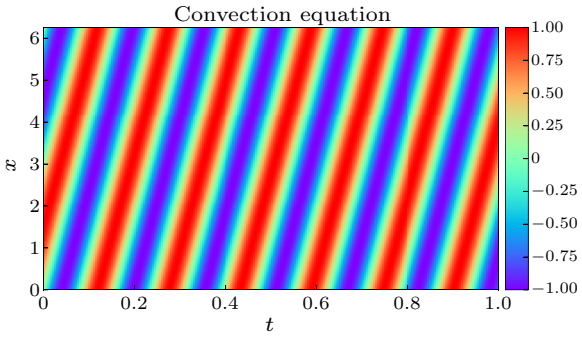


Fig. 20. The solution of the convection equation at $t = [0, 1]$.

We use two-interval PT-PINNs with intervals of [0,0.2], [0,0.6] and the final training in [0,1]. The prediction results and error plots are presented in Fig. 21.

It can be found from Table 6 that the CEICs significantly improve the performance of PT-PINNs for solving the convection equation, because in the progressive learning process in the time domain, CEICs can help PT-PINNs to obtain more accurate pseudo-labels.

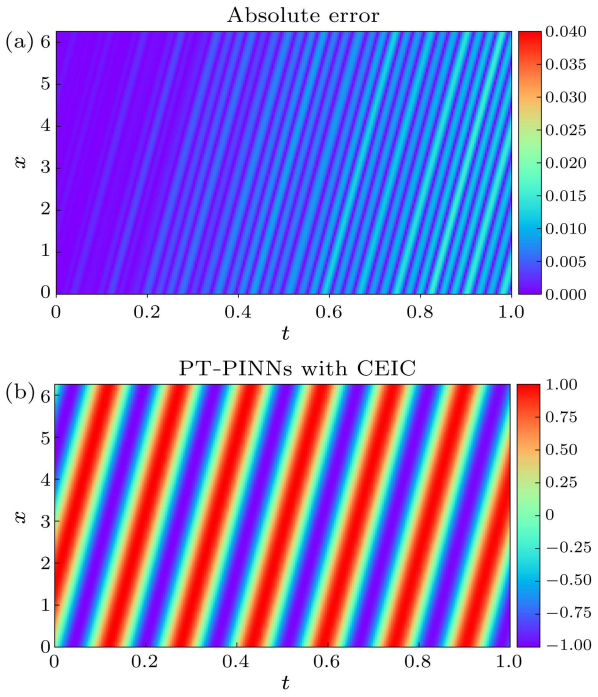


Fig. 21. The error (a) and the prediction result (b).

Table 6. Error comparison in convection equation.

Strategy	Relative loss
Standard PINNs	8.63×10^{-1}
PINNs with CEIC	4.21×10^{-1}
PINNs with HC	5.17×10^{-1}
PT-PINNs with CEIC	3.43×10^{-3}
PT-PINNs with HC	7.63×10^{-1}
PT-PINNs	6.89×10^{-3}
Seq2seq-PINNs	5.33×10^{-2}
Causal-PINNS	7.69×10^{-3}

5. Conclusion

In this paper, we discuss the significance of initial condition learning in PINNs and identify several challenges associated with employing surrogate neural networks to impose hard constraints on initial conditions. Furthermore, we propose a soft constraint strategy aimed at enhancing the learning of initial conditions within PINNs by introducing an additional loss term. This loss is composed of the derivative of the initial condition and the derivative of the neural network evaluated at the initial time. To strengthen these constraints, higher-order derivatives are incorporated into the loss function, along with a causal hyperparameter designed to maintain causal relationships during training. We refer to this approach as causally enhanced initial conditions (CEICs) in this paper. Experimental results demonstrate that CEICs can effectively improve PINNs' ability to learn initial conditions and can be seamlessly integrated with temporal marching strategies such as PT-PINNs to enhance overall learning performance.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (Grant Nos. 1217211 and 12372244).

References

[1] Lagaris I E, Likas A and Fotiadis D I 1998 *IEEE Trans. Neural Netw.* **9** 987
[2] Long Z, Lu Y and Dong B 2019 *J. Comput. Phys.* **399** 108925
[3] Liu Z, Yang Y and Cai Q 2019 *Appl. Math. Mech.-Engl. Ed.* **40** 237
[4] Irignano J and Spiliopoulos K 2018 *J. Comput. Phys.* **375** 1339
[5] Lu L, Jin P, Pang G, Zhang Z and Karniadakis G E 2021 *Nat. Mach. Intell.* **3** 218
[6] Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A and Anandkumar A 2021 *International Conference on Learning Representations*
[7] Raissi M, Perdikaris P and Karniadakis G E 2019 *J. Comput. Phys.* **378** 686
[8] Pang G, Lu L and Karniadakis G E 2019 *SIAM J. Sci. Comput.* **41** A2603
[9] Shen L H, Li D L, Zha W S, Li X and Liu X L 2022 *J. Pet. Sci. Eng.* **213** 110460
[10] Mao Z and Meng X 2023 *Appl. Math. Mech.-Engl. Ed.* **44** 1069
[11] Wang S F, Sankaran S and Perdikaris P 2024 *Comput. Methods Appl. Mech. Eng.* **421** 116813
[12] Gao Z, Tang T, Yan L and Zhou T 2024 *Commun. Appl. Math. Comput.* **6** 1720
[13] Lu L, Meng X, Mao Z and Karniadakis G E 2021 *SIAM Review* **63** 208
[14] Wu C, Zhu M, Tan Q, Kartha Y and Lu L 2023 *Comput. Methods Appl. Mech. Eng.* **403** 115671
[15] Yu J, Lu L, Meng X and Karniadakis G E 2022 *Comput. Methods Appl. Mech. Eng.* **393** 114823
[16] Wang S, Teng Y and Perdikaris P 2021 *SIAM J. Sci. Comput.* **43** A3055
[17] Jacot A, Gabriel F and Hongler C 2018 *Advances in Neural Information Processing Systems* **31** 8571
[18] Mattey R and Ghosh S 2022 *Comput. Methods Appl. Mech. Eng.* **390** 114474
[19] Wight C L and Zhao J 2020 arXiv:2007.04542v1 [math.NA]
[20] Krishnapriyan A, Gholami A, Zhe S, Kirby R and Mahoney M W 2021 *Advances in neural information processing systems* **34** 26548
[21] Guo J, Yao Y, Wang H and Gu T 2023 *J. Comput. Phys* **489** 112258
[22] Jagtap A D and Karniadakis G E 2020 *Commun. Comput. Phys.* **28** 2002
[23] Wang S, Yu X and Perdikaris P 2022 *J. Comput. Phys.* **449** 110768
[24] Penwarden M, Jagtap A D, Zhe S, Karniadakis G E and Kirby R M 2023 *J. Comput. Phys.* **493** 112464
[25] Guo J, Wang H F, Gu S L and Hou C P 2024 *Chin. Phys. B* **33** 050701
[26] Wang Q, Li D L and Zha W S 2025 *Phys. Fluids* **37** 013607
[27] Zha W, Li X, Xing Y, He L and Li D L 2020 *Advances in Geo-Energy Research* **4** 107
[28] Zha W, Li D and Lu Z 2018 *Advances in Geo-Energy Research* **2** 218
[29] Liu J N, Hou Q Z, Wei J G and Sun Z W 2023 *Chin. Phys. B* **32** 070702
[30] Tian S F, Li B and Zhao Z 2024 *Chin. Phys. Lett.* **41** 030201
[31] Peng K Y, Jing Y, Zhang W and Li J 2023 *Chin. Phys. B* **32** 040208