Engineering Applications of Artificial Intelligence 145 (2025) 110156

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Adversarial and self-adaptive domain decomposition physics-informed neural networks for high-order differential equations with discontinuities

Mingsheng Peng [a], Hesheng Tang [a,*], Yingwei Kou [a,b]

[a] Department of Disaster Mitigation for Structures, College of Civil Engineering, Tongji University, Shanghai, 200092, China
[b] China Shipping Environment Technology (Shanghai) Co., LTD, Shanghai, 200135, China

ARTICLE INFO

Keywords:
Physics-informed neural networks
Adversarial
Self-adaptive domain decomposition
Discontinuity
Multi-material problems
High-order problems

ABSTRACT

The technique of solving differential equations using physics-informed neural networks (PINNs) has received extensive attention and application. Analogous to the concept of adaptive mesh refinement in finite element methods, the PINNs framework should be tailored to the specific characteristics of the problem to further improve performance. However, the adaptive techniques in PINNs focus primarily on sampling and weighting, lacking the capability for adaptive domain decomposition. To address the limitations of PINNs in solving high-order problems with discontinuities, this paper proposes a novel type of adversarial and self-adaptive domain decomposition physics-informed neural networks (AS-PINNs). AS-PINNs leverage residual differences between subdomains to implement adversarial training among subnetworks, enabling the automatic and universal adjustment of subdomain interface positions and the capture of discontinuities with varying characteristics, representing a novel neural network approach. Compared to traditional domain decomposition methods, AS-PINNs eliminate the need for imposing tedious boundary and interface conditions through the loss function, particularly in high-order differential equations, thereby significantly reducing the complexity of the loss function and intrinsically improving accuracy. The results show that the self-adaptive adjustment of subdomain and the network structure makes AS-PINNs to be tailored to specific engineering problems, such as multi-material issues, high-order problems. As the order of the differential equations increases, the accuracy and speed advantages of AS-PINNs become more pronounced. For sixth-order differential equations, the solution speed of AS-PINNs is nine times faster compared to traditional domain decomposition PINNs. Code available at: https://github.com/Ning343/AS-PINNs.git.

1. Introduction

Deep learning have achieved significant breakthroughs in areas such as image recognition (Krizhevsky et al., 2012), cognitive science (Lake et al., 2015), genomics (Alipanahi et al., 2015), and scientific computing (Karniadakis et al., 2021). The integration of physical information broadens the application scenarios and interpretability of purely data-driven methods. Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) represent a mesh-free approach to solving differential equations, enhancing prediction accuracy and efficiency by incorporating physical equations into deep neural networks. Due to the mesh-free nature and the powerful nonlinear expressive ability of neural networks (Cybenko, 1989), PINNs show great potential in solving high-dimensional problems and complex boundary problems, and find extensive applications in forward problems (Faroughi et al., 2023; Vadeboncoeur et al., 2023; Wang et al., 2023; Wu et al., 2023), inverse problems (Bhowmick and Nagarajaiah, 2023; Li et al., 2022; Miao and Chen, 2023; Yu et al., 2021), operator learning (Hao et al., 2024; Kashefi et al., 2023; Lu et al., 2021a), and topology optimization (Jeong et al., 2023; Lu et al., 2021c). Despite these significant successes, some well-known drawbacks undeniably limit the further development and application of PINNs, such as low accuracy (Mattey and Ghosh, 2022; Wang et al., 2022b), lack of convergence guarantees (Saadat et al., 2022), and difficulty in hyperparameter tuning (Cao and Zhang, 2024; Wang et al., 2022a).

Due to the requirements of differentiation, PINNs typically use smooth and continuous activation functions. In engineering applications, the presence of heterogeneous materials, complex forces, and specific boundary and initial conditions often results in solutions that exhibit sharp discontinuities. Smooth activation functions hinder the

* Corresponding author.
E-mail address: thstj@tongji.edu.cn (H. Tang).

https://doi.org/10.1016/j.engappai.2025.110156
Received 14 November 2024; Received in revised form 25 December 2024; Accepted 22 January 2025
Available online 5 February 2025

precise approximation of discontinuous functions. High-order differentiation can expose singularities even in functions with weak discontinuities, potentially causing solution failures. Furthermore, discontinuities in high-order differential equations can cause interference between derivatives of different orders. This not only increases the computational burden but also makes it difficult to achieve the highest accuracy for all variables. After high-order differentiation or integration, significant error accumulation can severely limit the accuracy of the highest or lowest-order variables.

The network setup is the core of PINNs and should be adjusted according to the characteristics of the problem at hand. In the context of solving high-order differential equations with discontinuities, it is beneficial to adapt the network's configuration based on the current solution state and the information derived from it, in order to obtain more accurate results. A similar approach is employed in finite element methods, where the mesh configuration is adjusted according to the properties of the elastic body and external loading conditions. However, to date, neither the original nor the various improved versions of PINNs possess adaptive domain decomposition capabilities.

Existing research has not fully incorporated such adaptive strategies when addressing problems involving discontinuities and high-order equations. For problems involving steep gradients, increasing the number of residual points near the discontinuities (Hou et al., 2023; Lu et al., 2021b) enhances the attention of PINNs to discontinuities, thereby improving the accuracy of the solution. Modifying the network structure, as demonstrated by Chang et al. (2022) and Tseng et al. (2023), strengthens the network's ability to represent nonlinear behaviors. Reducing the weight assigned to singular value locations and employing weak forms to constrain the discontinuous points, thereby avoiding derivations at these points, effectively alleviates such issues (Antonio et al., 2024; Liu et al., 2023). Additionally, Reduced-order PINNs (RD-PINNs), which transform high-order differential equations into systems of lower-order equations, circumvent the need for repeated differentiation within a single network, thereby reducing computational overhead (Luong et al., 2024).

Another commonly used approach is domain decomposition, where the computational domain is pre-decomposed based on the locations of discontinuities, and the problem is solved accordingly. However, these methods rely on manually decomposing the computational domain, and the domain cannot be further adjusted during training based on the solution information. The interface conditions between subnetworks must be enforced through additional sampling and loss functions, which increases the complexity and difficulty of the solving process. In high-order differential equations, when discontinuities arise in a particular order of derivative, e.g., the primary variable or the flux, the remaining orders of quantities that adhere to continuity conditions are also subject to decomposition. This scenario results in a substantial rise in the number of loss function terms. The complexity is further compounded when derivatives of different orders experience discontinuities at different locations. Jagtap and Karniadakis (2020), which possess powerful representation and parallelization capabilities. Diao et al. (2023) successfully solved multi-material problems in solid mechanics using a domain decomposition approach and explored the potential optimal network framework. Hu et al. (2022) proposed augmented network structures that replace multiple networks with increased dimensions, thereby decreasing the total number of networks required. Sarma et al. (2024) proposed using the same parameters with different activation functions in different subnetworks, significantly reducing the number of parameters required for training. Despite the significant success of these efforts, the interface conditions rely on inefficient manual handling and additional residual points,

Research on self-adaptive PINNs primarily focuses on adaptive sampling (Hou et al., 2023; Wu et al., 2023; Yu et al., 2022), adaptive activation functions (Wang et al., 2023), and adaptive weights (McClenny and Braga-Neto, 2023; Subramanian et al., 2023), while lacking techniques for adaptive domain decomposition. The primary approach involves constructing functions based on residual information, such as adaptive sampling depending on the magnitude of residuals or using NKT theory (Wang et al., 2022a) for weight updating in the loss function. Another approach involves embedding specific trainable parameters within the loss function. PINNs update these parameters based on gradient descent to implement adaptive methods automatically, such as updating the weights of residual points, activation functions, and loss function terms (Kendall et al., 2018). The most relevant existing research focuses on adaptively adjusting the weighting coefficients for the summation of subnetworks, but these subnetworks still solve the problem over the entire domain, failing to address the singularities caused by discontinuities (Hu et al., 2023). Furthermore, some discontinuities only emerge over time (Bonkile et al., 2018), and the locations of parameter discontinuities are often unclear when solving inverse problems. Methods using prior knowledge for domain decomposition may become ineffective. Therefore, it is necessary to develop techniques for automatic localization of discontinuities.

The Adversarial and Self-Adaptive Domain Decomposition Physics-Informed Neural Networks (AS-PINNs) proposed in this paper adjust the network configuration based on the solution state and information, thereby solving high-order differential equations with discontinuities. The specific contributions of this paper are summarized as follows.

a) The mutual influence of variables of different orders in high-order differential equations with discontinuities is revealed. Specifically, the singularity caused by the discontinuity and its impact on PINNs is emphasized, with this impact affecting different orders of variables through different loss function terms. Moreover, the influence of this singularity on higher-order or lower-order variables differs.
b) A completely new PINNs computational framework is proposed. Unlike classical domain decomposition approaches that rely heavily on manual selection of decomposition schemes, AS-PINNs automatically capture discontinuity positions and adaptively adjust the domain decomposition scheme through the competition of subnetworks. The core innovation lies in the characteristic of neural networks to minimize the loss function. During training, each subnetwork tends to minimize its own loss function, resulting in adversarial competition at the interfaces. This competitive process drives the interfaces to dynamically adapt, ensuring that each subnetwork achieves the best possible local solution. As a result, the overall loss function is minimized in a coordinated manner, enabling automatic optimization of the domain decomposition scheme and achieving a balance between the subnetworks. In comparison, traditional domain decomposition methods require predefined or fixed interfaces, which lack flexibility when dealing with complex problems involving discontinuities.
c) Traditional PINNs relying on residual points or penalty-based loss functions face challenges in enforcing interface constraints, especially due to the mobility of interfaces in adaptive domain decomposition. To address this, the interface condition hard constraint method is proposed, which ensures that continuity conditions across subnetwork interfaces are consistently satisfied throughout the training process. This approach significantly reduces the complexity of setup and alleviates training difficulties. Compared to traditional domain decomposition methods, AS-PINNs achieve a ninefold speed improvement in solving sixth-order differential equations.

Through these advancements, AS-PINNs not only capture steep gradients and discontinuities with varying characteristics but also dynamically adjust sub-domains within a unified framework. This adaptive capability establishes AS-PINNs as a novel computational method that overcomes the limitations of existing domain decomposition and reduced-order approaches in solving complex high-order differential equations with discontinuities.

The remainder of this paper is organized as follows. In Section 2, after introducing the algorithm of PINNs, the extension to AS-PINNs is

presented. Section 3 systematically compares the performance of PINNs, AS-PINNs, RD-PINNs, and XPINNs for discontinuous function fitting. Section 4 demonstrates the effectiveness of AS-PINNs in capturing discontinuities and performing self-adaptive domain decomposition for high-order problems. Finally, the paper concludes in Section 5.

2. Methodology

2.1. PINNs and RD-PINNs

To maintain generality, consider the following m-th order differential equation:

$$a_0\lambda_0(x)\frac{d^k u(x)}{dx^k}+\ldots+a_r\lambda_r(x)\frac{d^{k-r}u(x)}{dx^{k-r}}+\ldots+a_k\lambda_k(x)u(x)-q(x)=0 \quad (1)$$

where $r, k\subset\mathbb{N}$, $r<k$. $a_r\subset\mathbb{R}$ and is a constant. $\lambda_r(x)$ are the system parameters, and $q(x)$ is the source term, with $x\in[0,L]$. $\lambda_r(x)$ represents the system parameter, and $q(x)$ denotes the source term, such as external loads. In practical engineering problems, both system parameters and source terms often exhibit spatial variation characteristics. Therefore, $\lambda_r(x)$ and $q(x)$ are considered spatially varying parameters. The equation is denoted as $\mathscr{T}\left(x;u,\ldots,d^k u(x)/dx^k;\lambda\right)$ and is subject to the boundary condition

$$\mathscr{B}\mathscr{C}(x;\hat{u};\boldsymbol{\lambda})=0 \quad (2)$$

where $\boldsymbol{\lambda}=[\lambda_0,\lambda_2,\cdots,\lambda_k]$. $\hat{u}(t,x;\boldsymbol{\theta})$ is the output of the neural networks (NNs), $\boldsymbol{\theta}$ is the set of parameters for the networks. Throughout this paper, variables with a hat symbol represent the output of the neural network. Residual points are collected inside the computational domain ($\mathscr{T}_{\mathscr{T}}$) and on boundaries ($\mathscr{T}_{\mathscr{B}\mathscr{C}}$). The loss functions of PINNs are defined

$$\mathscr{L}(\boldsymbol{\theta};\mathscr{T})=w_{\mathscr{T}}\mathscr{L}_{\mathscr{T}}(\boldsymbol{\theta};\mathscr{T}_{\mathscr{T}})+w_{\mathscr{B}\mathscr{C}}\mathscr{L}_{\mathscr{B}\mathscr{C}}(\boldsymbol{\theta};\mathscr{T}_{\mathscr{B}\mathscr{C}}) \quad (3)$$

where $w_{\mathscr{T}}$ and $w_{\mathscr{B}\mathscr{C}}$ are the weight coefficients, and

$$\mathscr{L}_{\mathscr{T}}(\boldsymbol{\theta};\mathscr{T}_{\mathscr{T}})=\frac{1}{|\mathscr{T}_{\mathscr{T}}|}\sum_{x\in\mathscr{T}_{\mathscr{T}}}\left|\mathscr{T}\left(x;\hat{u},\cdots,d^k\hat{u}(x)\Big/dx^k;\boldsymbol{\lambda}\right)\right|^2 \quad (4)$$

$$\mathscr{L}_{\mathscr{B}\mathscr{C}}(\boldsymbol{\theta};\mathscr{T}_{\mathscr{B}\mathscr{C}})=\frac{1}{|\mathscr{T}_{\mathscr{B}\mathscr{C}}|}\sum_{x\in\mathscr{T}_{\mathscr{B}\mathscr{C}}}|\mathscr{B}\mathscr{C}(x;\hat{u};\boldsymbol{\lambda})|^2 \quad (5)$$

Using optimization algorithms such as Adam and L-BFGS, the parameter set $\boldsymbol{\theta}^*$ that minimizes $\mathscr{L}(\boldsymbol{\theta};\mathscr{T})$ is found. The reduced-order form of RD-PINNs for this problem is (Luong et al., 2023)

$$\begin{cases}\dfrac{dv_i}{dx}-v_{i+1}=0 \quad (i=0,1,2,\ldots,k-1)\\ \mathscr{T}(x;v_0,\cdots,v_k;\boldsymbol{\lambda})=0\end{cases} \quad (6)$$

where

$$\mathscr{T}(x;v_0,\cdots,v_k;\boldsymbol{\lambda})=a_0\lambda_0(x)v_k+\ldots+a_r\lambda_r(x)v_{k-r}+\ldots+a_k\lambda_k(x)v_0-q(x) \quad (7)$$

v_0 is equal to u, and v_i is the i-th derivative of u. The idea of RD-PINNs involves using distinct networks $\hat{v}_0(x;\boldsymbol{\theta})$, $\hat{v}_1(x;\boldsymbol{\theta})$, …, $\hat{v}_k(x;\boldsymbol{\theta})$ to approximate v_0, v_1, …, v_k respectively. Since each derivative function is approximated by the corresponding network, all boundary conditions can be implemented through hard constraints on specific networks, such as $\hat{v}_i(x;\boldsymbol{\theta})=\hat{v}_i(x;\boldsymbol{\theta})\cdot x\cdot(x-L)$, ensuring $\hat{v}_i$ satisfies x and x-L equal to zero. Thus, the loss function of RD-PINNs includes only $\mathscr{L}_{\mathscr{T}_i}$, with the form

$$\mathscr{L}(\boldsymbol{\theta};\mathscr{T})=\sum_{i=1}^{k+1}w_{\mathscr{T}_i}\mathscr{L}_{\mathscr{T}_i}(\boldsymbol{\theta};\mathscr{T}_{\mathscr{T}_i}) \quad (8)$$

where $w_{\mathscr{T}_i}$ is the weight of $\mathscr{L}_{\mathscr{T}_i}$, and

$$\begin{cases}\mathscr{L}_{\mathscr{T}_i}=\dfrac{1}{N_{\mathscr{T}_i}}\sum\limits_{x\in\mathscr{T}_{\mathscr{T}_i}}w_{\mathscr{T}_i}\left|\dfrac{d\hat{v}_{i-1}}{dx}-\hat{v}_i\right|^2 \quad (i=1,2,\ldots,k)\\ \mathscr{L}_{\mathscr{T}_{k+1}}=\dfrac{1}{N_{\mathscr{T}_{k+1}}}\sum\limits_{x\in\mathscr{T}_{\mathscr{T}_{k+1}}}w_{\mathscr{T}_{k+1}}|\mathscr{T}(x;\hat{v}_0,\cdots,\hat{v}_k;\boldsymbol{\lambda})|^2\end{cases} \quad (9)$$

The primary advantage of RD-PINNs is their ability to unify the characteristics of the optimization task, which avoids the appearance of boundary loss functions.

2.2. AS-PINNs

2.2.1. Modified network structure

Inspired by mesh partitioning in finite elements, which adjusts based on the shape of the elastomer and the load application, an optimal solution strategy is to adjust the network settings of PINNs according to the characteristics of the problem. The principle of AS-PINN is illustrated in Fig. 1. In PINNs, the "discontinuity condition" specifically applies to u. Instead, each order of the derivative in AS-PINNs has its own distinct discontinuity characteristics, such as C^0 discontinuity, C^1 discontinuity, and so on. Functions with C^0 continuity but discontinuities in high-order derivatives, are considered weak discontinuities, meaning the function itself is continuous, but its derivatives exhibit discontinuities. On the other hand, functions with C^0 discontinuity are considered strong discontinuities, where the function itself experiences a jump, and its left and right limits are not equal.

Fig. 1 (a) illustrates the problem this paper aims to address, i.e., the primary variable or certain derivatives are smooth and continuous, while the remaining derivatives exhibit discontinuities with different characteristics. The solving mode using a single network encounters significant difficulty.

Fig. 1 (b) demonstrates the adversarial and self-adaptive domain decomposition principle of AS-PINNs. Specifically, for v_i that may have discontinuities, define a set of n trainable parameters $\mathbf{S}_i^{\{1\}}$ to capture the positions of discontinuities, where $\mathbf{S}_i^{\{1\}}=\{s_i^{\{1\}},\ldots,s_i^{\{j-1\}},s_i^{\{j\}},\ldots,s_i^{\{n\}}\}$, $n\in\mathbb{N}$, satisfying $s_i^{\{j-1\}}<s_i^{\{j\}}$. $s_i^{\{j\}}$ is denoted as the transition point. Since the number discontinuity points are unknown, n is initialized based on problem characteristics. The computational domain is divided into $n+1$ regions, $[0,\ s_i^{\{1\}})$, …, $[s_i^{\{j-1\}},\ s_i^{\{j\}})$, …, $[s_i^{\{n\}},\ L]$, forming the non-overlapping subdomains $\Omega_i^{\{j\}}$, where $\Omega_i=\cup_j^{n+1}\Omega_i^{\{j\}}$. Define the networks $\hat{v}_i^{\{1\}}$, …, $\hat{v}_i^{\{j\}}$, …, $\hat{v}_i^{\{n+1\}}$ for approximation of v_i in subdomains $\Omega_i^{\{1\}}$, …, $\Omega_i^{\{j\}}$, …, $\Omega_i^{\{n+1\}}$, forming

$$\hat{v}_i(x;\boldsymbol{\theta})=\sum_{j=1}^{j}\hat{v}_i^{\{j\}}(x;\boldsymbol{\theta})\cdot\mathbb{G}_{\Omega_i^{\{j\}}}(x) \quad (10)$$

where the operator $\mathbb{G}_{\Omega_i^{\{j\}}}(x)$ is defined as

$$\mathbb{G}_{\Omega_i^{\{j\}}}(x)=\begin{cases}0 & x\notin\Omega_i^{\{j\}}\\ 1 & x\in\Omega_i^{\{j\}}\end{cases} \quad (11)$$

Once the parameter set $\mathbf{S}_i^{\{1\}}$ is stable after a certain number of iterations, add $s_i^{\{n+1\}}$, forming the second iteration of the trainable parameter set $\mathbf{S}_i^{\{2\}}=\{s_i^{\{1\}},\ldots,s_i^{\{j-1\}},s_i^{\{j\}},\ldots,s_i^{\{n\}},s_i^{\{n+1\}}\}$. The network framework also gets updated. When $\mathbf{S}_i^{\{2\}}$ becomes stable, compare $s_i^{\{n+1\}}$ with each parameter in $\mathbf{S}_i^{\{1\}}$. If either

$$s_i^{\{n+1\}}\cong s_i^{\{j\}} \quad s_i^{\{j\}}\in\mathbf{S}_i^{\{1\}} \quad (12)$$

or

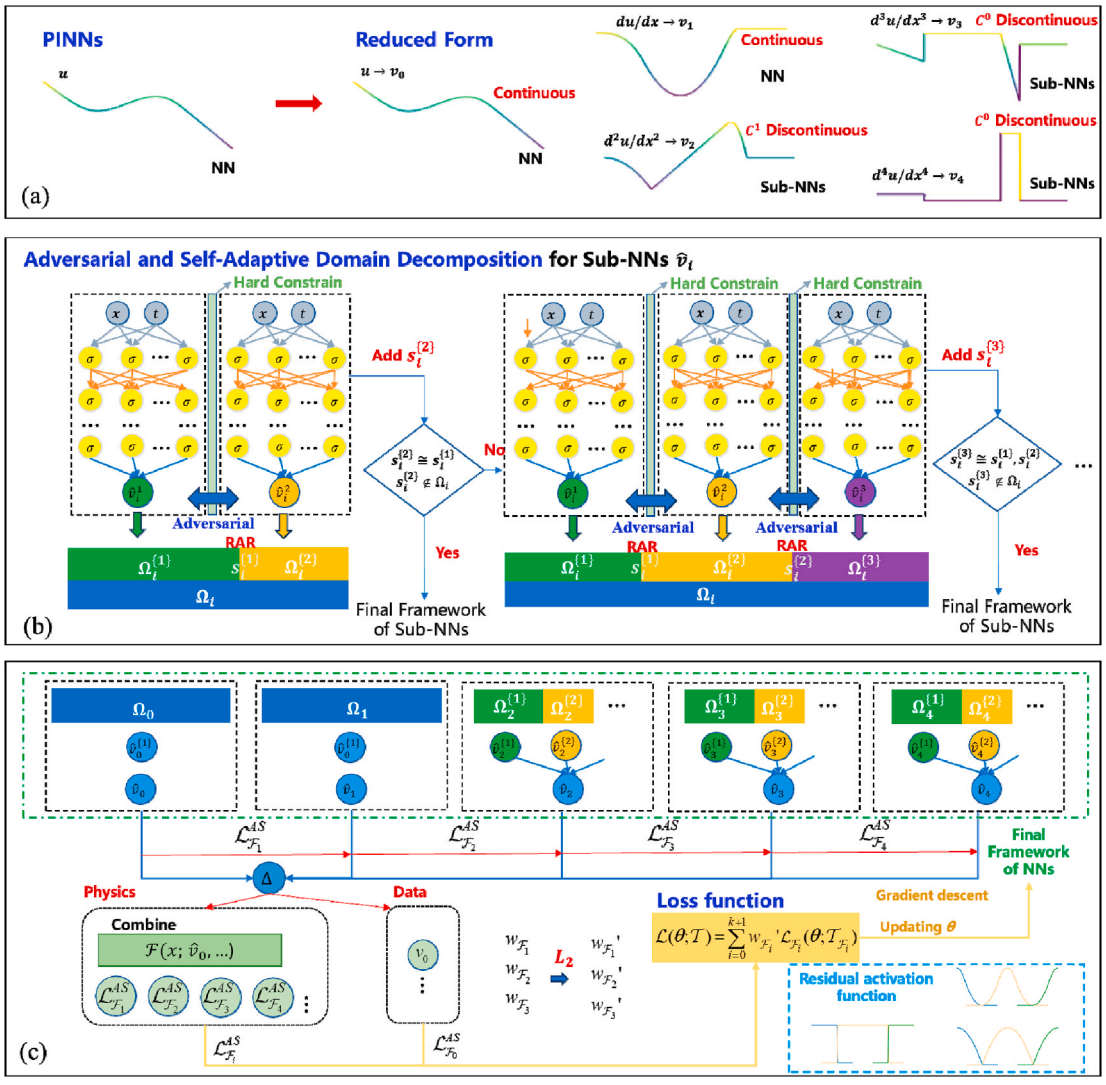


Fig. 1. The schematic of AS-PINNs. (a) The reduced-order form of the function; (b) Self-adaptive domain decomposition achieved through adversarial competition among subnetworks; RAR stands for Residual-based Adaptive Sampling. (c) The subnetworks formed after self-adaptive domain decomposition combine to create the final network framework used for loss function computation.

$$s_i^{\{n+1\}} \notin \Omega_i \tag{13}$$

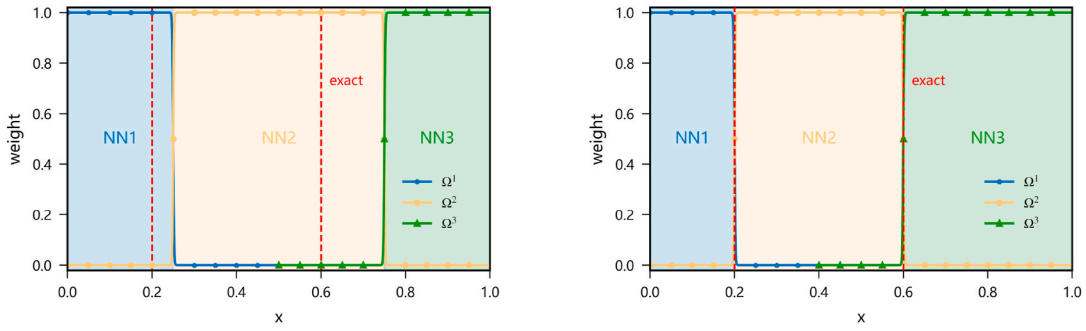
is satisfied, $s_i^{\{n+1\}}$ is considered invalid, indicating that the domain decomposition by $\mathbf{S}_i^{\{1\}}$ represents the optimal scheme. Otherwise, the newly added $s_i^{\{n+1\}}$ is regarded as a valid transition point and a new transition point. $s_i^{\{n+2\}}$ is added to search for potential discontinuities and update the domain decomposition scheme and network framework. Repeating these steps facilitates the automatic determination of the final optimal domain decomposition scheme and network framework.

Fig. 1 (c) illustrates the solution flowchart under the reduced-order framework. Different subnetwork frameworks approximate the primary variable and its various derivative functions: complex derivatives with discontinuities are approximated using multiple subnetworks in each subdomain, while smooth derivatives are approximated using a single subnetwork. The solution mode under the reduced-order framework unifies the loss function form and ensures the network fully considers the difficulty of fitting tasks for each derivative order, maximizing the use of solution information to adjust domain decomposition scheme. Using $\mathscr{L}^{AS}$ to denote the loss function of AS-PINNs, $\mathscr{L}^{AS}_{\mathscr{T}_i}$ represents the loss function term related to the network $\hat{v}_i$. Different neural networks are combined through the residuals of $\mathscr{L}^{AS}_{\mathscr{T}_i}$ to form the final loss function. Combining physical information and data to construct the loss function, the gradient descent algorithm updates network parameters. To provide structural clarity, Fig. 1 illustrates a series of subnetworks. In practice, it is possible to use a single larger neural network with shared inputs to output multiple variables. A single variable output can be regarded as a subnetwork. This approach not only reduces network complexity but also achieves competitive results. Similar strategies have been adopted in existing studies (Diao et al., 2023; Lu et al., 2021a).

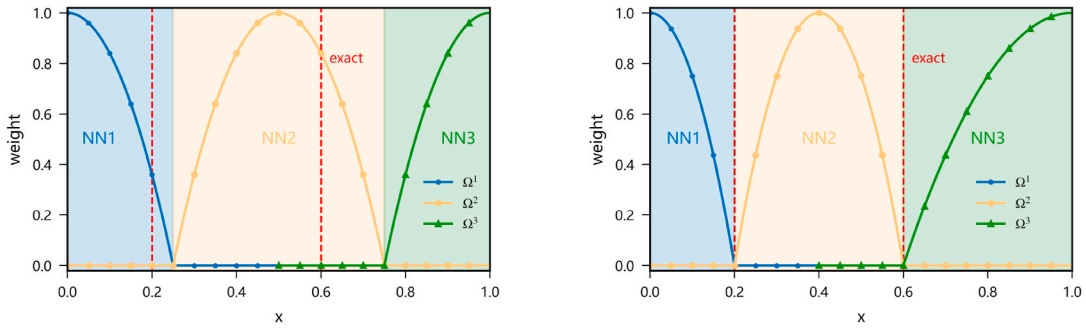
Since the solution involves only the first-order derivative function of the network, if v_i exhibits C^1 continuity, a single subnetwork can be used for approximation without singularity arising from differential operations. Considering the case where only v_i exhibits C^1 discontinuous while v_{i+1} exhibits C^0 discontinuous, the loss function for AS-PINN is as follows

$$\mathscr{L}^{AS}(\boldsymbol{\theta};\mathscr{T}) = \sum_{i=1}^{k+1} \boldsymbol{w}_{\mathscr{T}_i} \mathscr{L}^{AS}_{\mathscr{T}_i}\left(\boldsymbol{\theta};\mathscr{T}_{\mathscr{T}_i};\mathbf{S}_i^{\{j\}}\right) \tag{14}$$

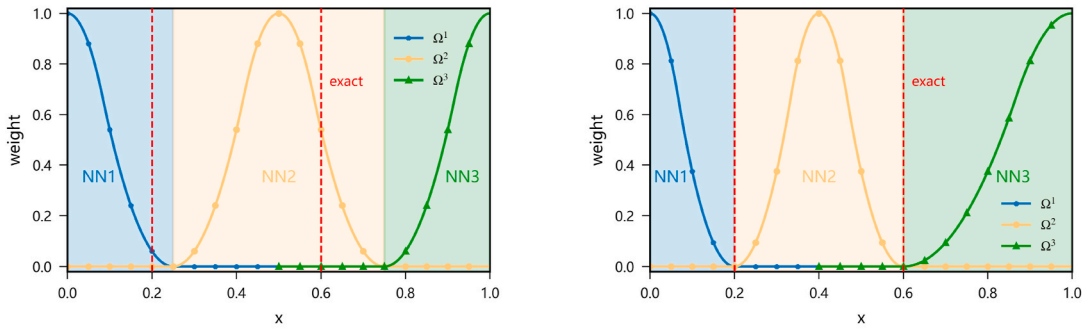
where



(a) $\mathrm{H}_{\Omega_i^{(j)}}^{\{1\}}(x)$



(b) $\mathrm{H}_{\Omega_i^{(j)}}^{\{2\}}(x)$



(c) $\mathrm{H}_{\Omega_i^{(j)}}^{\{3\}}(x)$

Fig. 2. Residual activation function in spatial domain. (First column) the initial domain decomposition scheme. (Second column) the domain decomposition scheme after training completion.

$$\begin{cases} \mathscr{L}^{AS}_{\mathscr{T}_1} = \dfrac{1}{N_{\mathscr{T}_1}} \sum\limits_{x\in\mathscr{T}_{\mathscr{T}_1}} w_{\mathscr{T}_1} \left| \dfrac{d^{n_0}\widehat{v}_0}{dx^{n_0}} - \widehat{v}_1 \right|^2 \\ \vdots \\ \mathscr{L}^{AS}_{\mathscr{T}_i} = \dfrac{1}{N_{\mathscr{T}_i}} \sum\limits_{x\in\mathscr{T}_{\mathscr{T}_i}} w_{\mathscr{T}_i} \left| \dfrac{d^{n_{i-1}}\widehat{v}_{i-1}}{dx^{n_{i-1}}} - \sum\limits_{j=1}^{n+1} \widehat{v}_i^{\{j\}} \cdot \mathbb{G}_{\Omega_i^{\{j\}}} \right|^2 \\ \mathscr{L}^{AS}_{\mathscr{T}_{i+1}} = \dfrac{1}{N_{\mathscr{T}_{i+1}}} \sum\limits_{x\in\mathscr{T}_{\mathscr{T}_{i+1}}} w_{\mathscr{T}_{i+1}} \left| \sum\limits_{j=1}^{n+1} \dfrac{d^{n_i}\widehat{v}_i^{\{j\}}}{dx^{n_i}} \cdot \mathbb{G}_{\Omega_i^{\{j\}}} - \widehat{v}_{i+1} \right|^2 \\ \vdots \\ \mathscr{L}^{AS}_{\mathscr{T}_k} = \dfrac{1}{N_{\mathscr{T}_k}} \sum\limits_{x\in\mathscr{T}_{\mathscr{T}_k}} w_{\mathscr{T}_k} \left| \dfrac{d^{n_{k-1}}\widehat{v}_{k-1}}{dx^{n_{k-1}}} - \widehat{v}_k \right|^2 \\ \mathscr{L}^{AS}_{\mathscr{T}_{k+1}} = \dfrac{1}{N_{\mathscr{T}_{k+1}}} \sum\limits_{x\in\mathscr{T}_{\mathscr{T}_{k+1}}} w_{\mathscr{T}_{k+1}} \left| \mathscr{F}\left(x; \widehat{v}_0, \cdots, \widehat{v}_i^{\{j\}}, \cdots, \widehat{v}_k; \boldsymbol{\lambda}\right) \right|^2 \end{cases} \tag{15}$$

only $\mathscr{L}^{AS}_{\mathscr{T}_i}$ and $\mathscr{L}^{AS}_{\mathscr{T}_{i+1}}$ are changed compared to Equation (9).

2.2.2. Self-adaptive domain decomposition method for sub-networks

The operator $\mathbb{G}_{\Omega_i^{\{j\}}}$ in equation (10) is a Boolean function that activates the network and cannot update interface locations through gradient descent of the loss function. Inspired by window functions and compactly supported wavelet functions in wavelet transform (Daubechies, 1992; Liu et al., 2020), the network activation operator $\mathbb{G}_{\Omega_i^{\{j\}}}$ is converted into the corresponding residual activation operator $\mathbb{H}_{\Omega_i^{\{j\}}}$. $\mathscr{L}^{AS}_{\mathscr{T}_i}$ and $\mathscr{L}^{AS}_{\mathscr{T}_{i+1}}$ are modified as

$$\begin{cases} \mathscr{L}^{AS}_{\mathscr{T}_i} = \dfrac{1}{N_{\mathscr{T}_i}} \sum\limits_{j=1}^{n+1} \mathbb{H}_{\Omega_i^{\{j\}}} \cdot \sum\limits_{x\in\mathscr{T}_{\mathscr{T}_i}} w_{\mathscr{T}_i} \left| \dfrac{d^{n_{i-1}}\widehat{v}_{i-1}}{dx^{n_{i-1}}} - \widehat{v}_i^{\{j\}} \right|^2 \\ \mathscr{L}^{AS}_{\mathscr{T}_{i+1}} = \dfrac{1}{N_{\mathscr{T}_i}} \sum\limits_{j=1}^{n+1} \mathbb{H}_{\Omega_{i+1}^{\{j\}}} \cdot \sum\limits_{x\in\mathscr{T}_{\mathscr{T}_i}} w_{\mathscr{T}_{i+1}} \left| \dfrac{d^{n_i}\widehat{v}_i^{\{j\}}}{dx^{n_i}} - \widehat{v}_{i+1} \right|^2 \end{cases} \tag{16}$$

$\mathbb{H}_{\Omega_i^{\{j\}}}$ is defined as

$$\mathbb{H}_{\Omega_i^{\{j\}}} = \begin{cases} 0 & x \notin \Omega_i \\ h(x) & x \in \Omega_i \end{cases} \tag{17}$$

To enable the update of $\mathbf{S}_i$ through backpropagation, $\mathbb{H}_{\Omega_i^{\{j\}}}$ cannot be defined as an indicator function but must satisfy continuity. Taking $n = 2$, $L = 1$ as an example, three possible regularized forms of $\mathbb{H}_{\Omega_i^{\{j\}}}$ are shown in Fig. 2. $s_i^{\{1\}} = 0.25$, $s_i^{\{2\}} = 0.75$,In the left column of Fig. 2, the domain decomposition scheme is initialized with boundaries $s_i^{\{1\}} = 0.25$ and $s_i^{\{2\}} = 0.75$. After training, as shown in the right column, the subdomain boundaries $s_i^{\{1\}}$ and $s_i^{\{2\}}$ are successfully adjusted and localized to the positions of the discontinuities at $s_i^{\{1\}} = 0.2$ and $s_i^{\{2\}} = 0.6$. AS-PINNs adopt trainable and freely movable window functions, allowing the method to locate discontinuities and achieve self-adaptive domain decomposition.

Denote $s_i^{\{0\}} = -s_i^{\{1\}}$ and $s_i^{\{n+1\}} = 2L - s_i^{\{n\}}$. $\mathbb{H}^{\{1\}}_{\Omega_i^{\{j\}}}(x)$ is defined as

$$\mathbb{H}^{\{1\}}_{\Omega_i^{\{j\}}}(x) = 1 - Sigmoid\left(p\left(x - s_i^{\{j-1\}}\right)\right) - Sigmoid\left(p\left(-x + s_i^{\{j\}}\right)\right) \tag{18}$$

By controlling the size of p, the steepness of the function can be adjusted. When p is sufficiently large, the function value outside the defined region is considered to be zero. $\mathbb{H}^{\{2\}}_{\Omega_i^{\{j\}}}(x)$ is defined as

$$\mathbb{H}^{\{2\}}_{\Omega_i^{\{j\}}}(x) = \frac{4}{\left(s_i^{\{j\}} - s_i^{\{j-1\}}\right)^2} \cdot Relu\left(x - s_i^{\{j-1\}}\right) \cdot Relu\left(-x + s_i^{\{j\}}\right) \tag{19}$$

Unlike $\mathbb{H}^{\{1\}}_{\Omega_i^{\{j\}}}(x)$, even if p is very small, the function value of $\mathbb{H}^{\{2\}}_{\Omega_i^{\{j\}}}(x)$ outside the defined region is strictly zero. $\mathbb{H}^{\{3\}}_{\Omega_i^{\{j\}}}(x)$ is constructed

$$\begin{aligned} \mathbb{H}^{\{2\}}_{\Omega_i^{\{j\}}}(x) = p\Bigg[& Relu\left(x - s_i^{\{j-1\}}\right)^2 - 3Relu\left(x - \frac{2}{3}s_i^{\{j-1\}} - \frac{1}{3}s_i^{\{j\}}\right)^2 \\ & + 3Relu\left(x - \frac{1}{3}s_i^{\{j-1\}} - \frac{2}{3}s_i^{\{j\}}\right)^2 - Relu\left(x - s_i^{\{j\}}\right)^2 \Bigg] \end{aligned} \tag{20}$$

$\mathbb{H}^{\{3\}}_{\Omega_i^{\{j\}}}(x)$ remains continuous in its derivatives, except where it strictly satisfies zero outside the defined domain.

The residual activation function has a weight of zero at the transition point, with the weight gradually increasing as it moves away from the transition point. As training progresses, each subnetwork minimizes its loss function within its respective subdomain in a manner similar to an adversarial approach, leading to an improved domain decomposition scheme that effectively captures discontinuities and enables self-adaptive decomposition.

To better explain the principle of self-adaptive domain decomposition in AS-PINNs, detailed and general illustration was provided in Fig. 3. The blue line represents the residual activation function used to activate NN1, and the green line represents the residual activation function used to activate NN2. The gray line depicts the spatial distribution of the residuals (loss function). For each individual subnetwork, the network aims to minimize the loss function within its respective computational domain. As a result, the residual activation function for NN1 tends to shift left, while the residual activation function for NN2 shifts right. This leads to a competitive interaction at the interface positions. When the residuals in the regions of NN1 and NN2 become comparable, the competition between the two subnetworks becomes balanced, and the transition point stabilizes. This is shown in Fig. 3 (a), where the sizes of the blue and green arrows are equal, indicating no dominant "competitiveness" between the two subnetworks. However, when the residual in the region of NN2 is noticeably larger than that of NN1, the "competitiveness" of NN2 will exceed that of NN1, and the interface position will shift towards the right. This is illustrated in Fig. 3 (b), where the green arrow is larger than the blue arrow, signifying the dominance of NN2.

The accuracy of transition point localization largely relies on the information from residual points. The domain decomposition scheme is dynamically adjusted based on the loss functions of the residual points. The magnitude of the residuals at training points reflects the difficulty of the solution at that location and provides critical information about the current state of the solution. Initially, residual points are distributed with a coarse resolution, which may not accurately localize discontinuities due to insufficient density in the regions where singularities exist, as shown in Fig. 3 (b). The red initialization training points fail to effectively capture the anomalous values of the loss function near $x = 0.8$. This limits the precision of transition point localization at the early stages of training. To address this, AS-PINNs adopt Residual-based Adaptive Sampling (RAR) method. Instead of globally increasing the density of training points, the algorithm automatically concentrates more residual points in the vicinity of discontinuities, as indicated by the blue scattered points. This localized refinement provides additional critical information for accurately capturing the discontinuity at a finer resolution while maintaining computational efficiency. The transition points are considered stabilized when the changes in their positions become negligible, indicating convergence of the solution. Transition points are considered accurately localized when their positions converge to the actual locations of discontinuities.

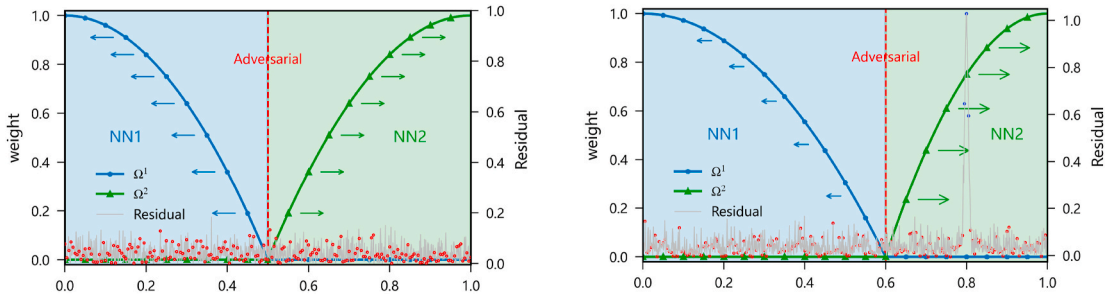


Fig. 3. The subnetworks aim to minimize their respective computational domains to reduce their individual loss functions, thereby creating competition at the interface. (First column) The competition between the two subnetworks is balanced as there is no significant difference in the residuals of their respective regions. (Second column) The right subnetwork exhibits greater competitiveness compared to the left one, as the left region has a larger residual. The red scatter points represent the initially sampled training points, while the blue scatter points are the adaptively sampled training points. The size of the arrows indicates the level of competitiveness.

Due to the continuous movement of subnetwork interfaces, enforcing continuity conditions through the loss function and residual points becomes challenging. To address this, a hard constraint method is employed for interface conditions.

Given that v_i satisfies the C^0 continuity condition, $v_i(0) = 0$ and $v_i(L) = 0$. Using the idea of Lagrangian interpolation, construct the trainable interpolation function set $\mathbf{C} = \{c_1, \ldots, c_j, \ldots, c_n\}$. Denote the output of the network $\widehat{v}_i^j(x;\boldsymbol{\theta})$ as $\mathscr{N}_i^{\{j\}}(x;\boldsymbol{\theta})$. Let

$$\widehat{v}_i^{\{j\}}(x;\boldsymbol{\theta}) = \begin{cases} \mathscr{N}_i^{\{j\}}(x;\boldsymbol{\theta})\cdot x\cdot(x-s_j) + v_i(0)\cdot\dfrac{(x-s_j)}{(-s_j)} \\ \qquad + c_j\cdot\dfrac{x}{s_j}, j = 1 \\ \mathscr{N}_i^{\{j\}}(x;\boldsymbol{\theta})\cdot(x-s_{j-1})\cdot(x-s_j) + c_{j-1}\cdot\dfrac{(x-s_j)}{(s_{j-1}-s_j)} \\ \qquad + c_j\cdot\dfrac{(x-s_{j-1})}{(s_j-s_{j-1})}, j = 2,\ldots,n \\ \mathscr{N}_i^{\{j\}}(x;\boldsymbol{\theta})\cdot(x-s_{j-1})\cdot(x-L) + c_{j-1}\cdot\dfrac{(x-L)}{(s_{j-1}-L)} \\ \qquad + v_i(L)\cdot\dfrac{(x-s_{j-1})}{(L-s_{j-1})}, j = n+1 \end{cases} \tag{21}$$

Through the setting in equation (21), despite the continuous movement of the network interface, C^0 continuity conditions of the network will always be satisfied. When solving high-dimensional differential equations, it is only necessary to replace the parameter set $\mathbf{C} = \{c_1, \ldots, c_j, \ldots, c_n\}$ with the corresponding network set, i.e., let $\mathscr{C}_j = \mathscr{N}_i^{\{C_j\}}(x_2, x_3, \ldots;\boldsymbol{\theta})$, where $x_2, x_3, \ldots$ are the independent variables along the discontinuity.

Algorithm 1. The AS-PINN for solving high-order differential equations with discontinuities

Initialize n, the potential lower bound for the number of discontinuity points
Set $m = 1$
While True:
Step 1: Set parameters and configure the neural network
1.1 Construct the set of trainable transition points $\mathbf{S}_i^{\{m\}} = \{s_i^{\{1\}}, \ldots, s_i^{\{j-1\}}, s_i^{\{j\}}, \ldots, s_i^{\{n\}}\}$
1.2 Construct residual activate function
1.3 Construct neural networks with parameters $\theta^{\{m\}}$
1.4 Randomly sample training data within the domain
1.5 Define the loss function
Step 2: Train the neural network

(*continued on next column*)

(*continued*)

2.1 Perform weight updates with a higher learning rate and faster frequency. Rapidly approach a rough resolution space
2.2 Pause weight updates and collect training points at a slower frequency. Use smaller learning rates to refine breakpoints in fine resolution space
2.3 Save network parameters $\theta^{\{m\}}$ and the set $\mathbf{S}_i^{\{m\}}$
Step 3: Check termination condition
If $m! = 1$
If $s_i^{\{n\}} \cong s_i^{\{j\}}, s_i^{\{j\}} \in \mathbf{S}_i^{\{m-1\}}$ or $s_i^{\{n\}} \notin \Omega_i$
$n = n$ - 1
$m = m$ - 1
end
Reload $\mathbf{S}_i^{\{m\}}$ and network parameters $\theta^{\{m\}}$
Break
end
$n = n + 1$
$m = m + 1$
end

Using interface condition hard constrain method between subnetworks and hard BCs constraint method under the reduced-order model, the loss function of AS-PINNs will include only $\mathscr{L}_{\mathscr{F}_i}^{AS}$, which is the fitting residual of the network to the gradient information or the target function. The fitting residuals can be expressed using the L_2 error $\varepsilon(v, \widehat{v})$, a relative measure, defined as

$$\varepsilon(v, \widehat{v}) = \frac{\|v - \widehat{v}\|}{\|v\|} \tag{22}$$

and

$$\|v\| = \sqrt{\int_\Omega v^2 dx} \tag{23}$$

The absolute error in the loss function of PINNs, due to variations in magnitude, often results in loss function terms being on different scales. Therefore, the absolute error can be replaced by the L_2 error, or the weight of the loss function can be adjusted using the L_2 error, thereby achieving a balance among different loss terms. Specifically, if the L_2 errors of current loss function terms are $\{L_2^{\{1\}}, \ldots L_2^{\{i\}}, \ldots\}$, the weights can be updated according to

$$w_{\mathscr{F}_i}{}' = L_2^{\{i\}} / \min\{L_2^{\{1\}}, \ldots L_2^{\{i\}}, \ldots\} \tag{24}$$

$$w_{\mathscr{F}_i}{}' = L_2^{\{i\}} / \max\{L_2^{\{1\}}, \ldots L_2^{\{i\}}, \ldots\} \tag{25}$$

Weight updates will make AS-PINNs focus on tasks with relatively low training accuracy, ensuring all tasks converge with similar accuracy. Additionally, discontinuities may be distributed across different orders

of derivatives, and improper weights $w_{\mathscr{T}_i}$ may prevent AS-PINNs from capturing discontinuities of certain derivatives which are not strong (e.g., C^0 continuity but a C^1 discontinuity). Dynamic adjustment of weights can further enhance the ability of AS-PINNs to capture discontinuities with various characteristics. The AS-PINN for solving high-order differential equations with discontinuities is summarized in Algorithm 1.

3. AS-PINNs for function approximation

Problems exposed by PINNs when fitting discontinuous functions reflect the inefficiency or failure in solving differential equations with discontinuous characteristics. To illustrate the impact of discontinuities on function fitting, the original PINNs (denoted as PINNs), RD-PINNs, XPINNs, and AS-PINNs are used to complete the following discontinuous function fitting task. Assuming the domain is $\Omega = [0, L]$, and the primary variable is

$$u = \begin{cases} 4x, & x \in [0, 0.4] \\ -x^2 + 4.8x - 0.16, & x \in (0.4, 0.9] \\ 3x + 0.65, & x \in (0.9, 2] \\ 2x + 2.65, & x \in (2, 3] \end{cases} \tag{26}$$

u has discontinuities at $d_1 = 0.4$, $d_2 = 0.9$, $d_3 = 2$. Considering noticeable discontinuities, the primary variable u has C^1 discontinuities, the first derivative u' has both C^0 and C^1 discontinuities, and the second derivative u'' has C^0 discontinuities. Given that u' and u'', as well as $u(0)$, are known, different methods require varying numbers of output variables. A single network is configured to output multiple variables. In different computational methods, 100 residual points are randomly sampled within the computational domain, employing 3 hidden layers with 50 neurons per layer in fully connected neural networks to output the required variables. Training is performed for 160,000 epochs. The Adam optimizer is employed for optimization, starting with a learning rate of 10^{-3} for the first 20,000 iterations, which is then reduced to 10^{-4}. All loss function weights are initialized at 1, with an additional 5 sampling points incorporated every 20,000 epochs using the RAR algorithm. Based on Equation (24), the weights in AS-PINNs and RD-PINNs are updated every 20,000 epochs. The computational experiments in this study were conducted using an NVIDIA GeForce RTX 4070 Ti, an Intel Core i5-13400F, and 32 GB of DDR5 RAM. DeepXDE (Lu et al., 2021b) was utilized to implement AS-PINNs in this paper, and it can be easily extended to higher-level GPUs.

AS-PINNs require setting up networks $\widehat{v}_0^{\{1\}}$, $\widehat{v}_0^{\{2\}}$, $\widehat{v}_0^{\{3\}}$, $\widehat{v}_0^{\{4\}}$ and $\widehat{v}_1^{\{1\}}$, $\widehat{v}_1^{\{2\}}$, $\widehat{v}_1^{\{3\}}$, $\widehat{v}_1^{\{4\}}$ to approximate u and du/dx in the subdomains $\Omega^{\{1\}} = [0, 0.4)$, $\Omega^{\{2\}} = [0.4, 0.9)$, $\Omega^{\{3\}} = [0.9, 2)$, $\Omega^{\{4\}} = [2, 3]$. Here $\Omega_0^{\{j\}} = \Omega_1^{\{j\}}$, the subscript is omitted. The loss function for AS-PINNs is

$$\begin{cases} \mathscr{L}^{AS}_{\mathscr{T}_1} = \dfrac{1}{N_{\mathscr{T}_1}} \displaystyle\sum_{j=1}^{n+1} \mathbb{H}_{\Omega^{\{j\}}} \sum_{x \in \mathscr{T}_{\mathscr{T}_1}} w_{\mathscr{T}_1} \left| \widehat{v}_1^{\{j\}} - \frac{du}{dx} \right|^2 \\ \mathscr{L}^{AS}_{\mathscr{T}_2} = \dfrac{1}{N_{\mathscr{T}_2}} \displaystyle\sum_{j=1}^{n+1} \mathbb{H}_{\Omega^{\{j\}}} \sum_{x \in \mathscr{T}_{\mathscr{T}_2}} w_{\mathscr{T}_2} \left| \frac{d\widehat{v}_1^{\{j\}}}{dx} - \frac{d^2u}{dx^2} \right|^2 \\ \mathscr{L}^{AS}_{\mathscr{T}_3} = \dfrac{1}{N_{\mathscr{T}_3}} \displaystyle\sum_{j=1}^{n+1} \mathbb{H}_{\Omega^{\{j\}}} \sum_{x \in \mathscr{T}_{\mathscr{T}_3}} w_{\mathscr{T}_3} \left| \frac{d\widehat{v}_0^{\{j\}}}{dx} - \widehat{v}_1^{\{j\}} \right|^2 \end{cases} \tag{27}$$

Initially, to assess the impact of various residual activation functions on discontinuity localization, a comparison of solution outcomes under different residual activation functions was performed. The iterative process of the transition points is illustrated in Fig. 4. Fig. 5 provides a detailed illustration of the self-adaptive domain decomposition and the adversarial process of subnetworks under $\mathbb{H}^{\{2\}}(x)$, while Fig. 6 presents the results obtained using various methods. Specific L_2 relative errors of

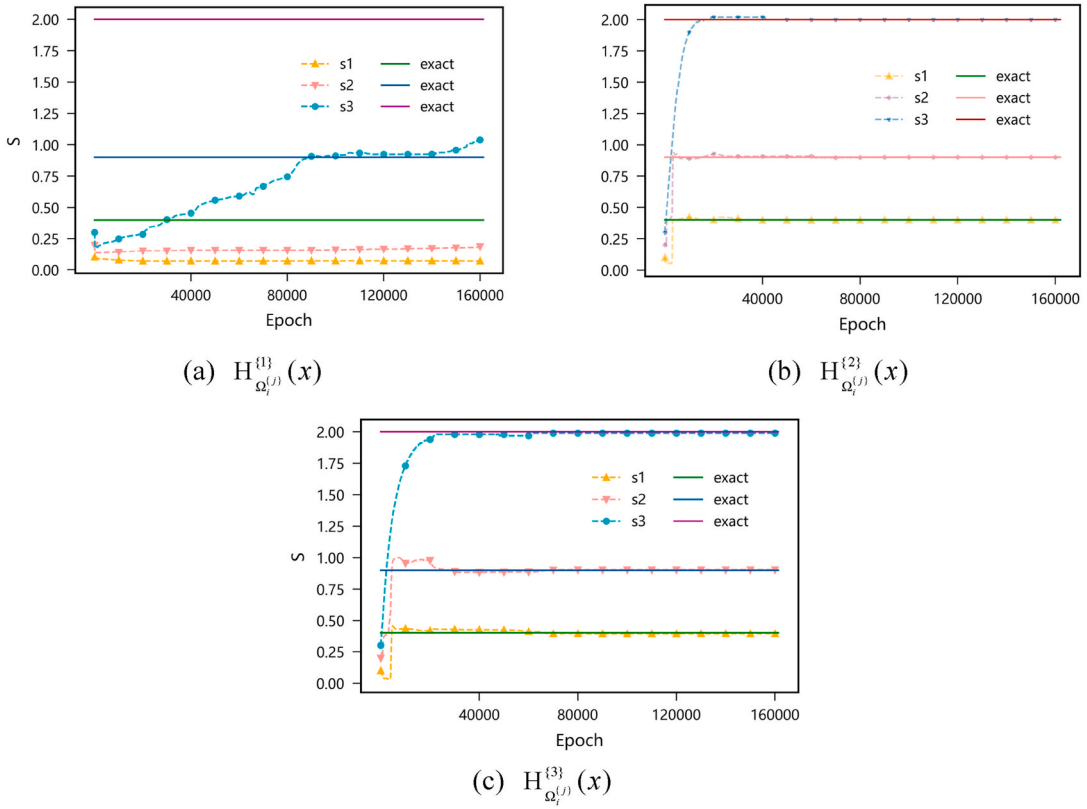


Fig. 4. The convergence of interface location of AS-PINNs using different residual activation functions. Transition points are intentionally distributed at one end of the computational domain.

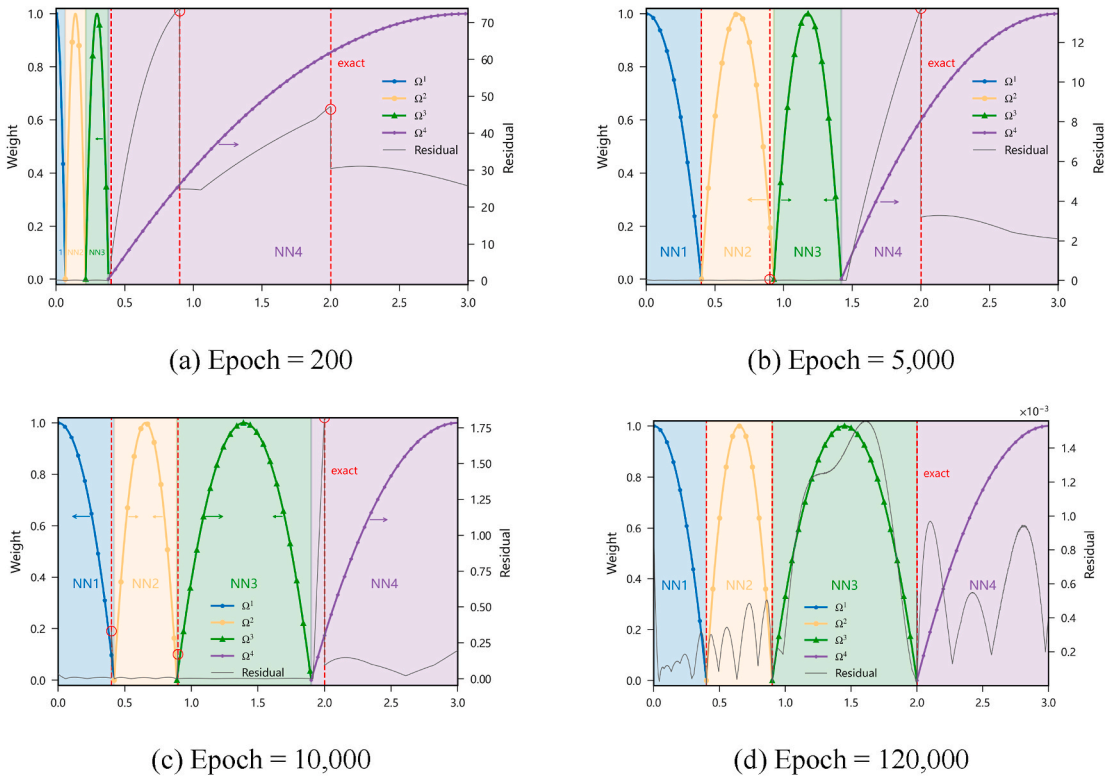


Fig. 5. Self-Adaptive domain decomposition process at different epochs under $\mathbb{H}^{\{2\}}(x)$. The size of the arrows indicates the level of competitiveness.

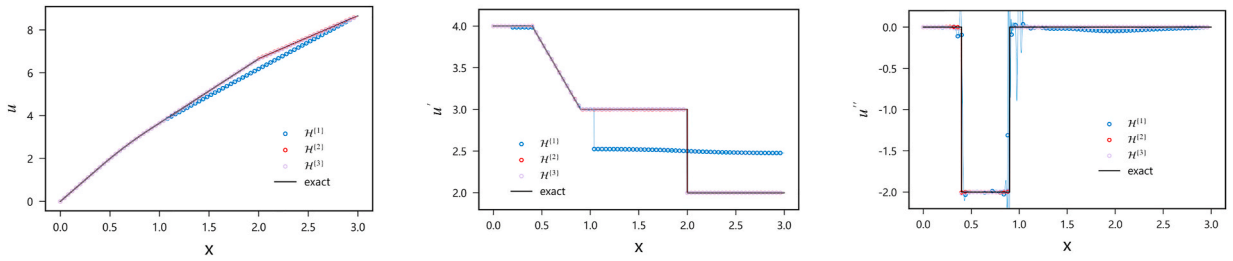
different methods and final positions of transition points are given in Table 1. In the table, the optimal results are underlined, while the second-best results are highlighted in bold. Although the residual activation function $\mathbb{H}^{\{1\}}(x)$, resembling a rectangular window, effectively activates residuals across the subdomain $\Omega_i^{\{j\}}$, the sharp transition at the interface introduces a large gradient, complicating the optimization of transition points during training. Moreover, regardless of the value of p in Equation (18), the residuals of network $\widehat{v}_i^{\{j\}}$ cannot be strictly zero outside the domain $\Omega_i^{\{j\}}$. Theoretically, the derivative of $\widehat{v}_i^{\{j\}}$ at the discontinuity point approaches infinity, and even if the weight at this point is relatively small, the extraneous information can still influence the solution, causing failure. These factors lead to the failure of transition points in accurately capturing the location of discontinuities.

$\mathbb{H}^{\{2\}}(x)$ and $\mathbb{H}^{\{3\}}(x)$ effectively capture the discontinuities, which are distributed throughout the entire domain, as shown in Fig. 4. To fully demonstrate the ability of AS-PINNs to capture discontinuity points, the transition points are deliberately initialized at one end of the domain. Some transition points need to cross one or two discontinuity points to reach the real discontinuity point location, forming the optimal network framework. Uniformly distributed initializations make it easier for transition points to capture discontinuities. Despite this, the transition points quickly obtain most of the discontinuity locations after training begins, combining this with further refinement through the RAR algorithm ultimately achieves precise capture of the discontinuities. The self-adaptive domain decomposition process of the four subnetworks is shown in Fig. 5. When the residuals in the subdomains are comparable, the subnetworks exhibit equal competitiveness at the interface. If one subdomain has a significantly larger residual (marked by the red circle), the corresponding subnetwork becomes more competitive, leading to a shrinkage of its subdomain. The size of the arrows indicates the level of competitiveness. Before 10,000 epochs, the residual value in the region where NN4 is located is larger, giving NN4 a stronger competitiveness during training. The interface position between NN4 and NN3 continually shifts to the right, eventually being located at $x = 2$. Similarly, through the competition caused by the residual differences, the interface positions between NN1 and NN2, and NN2 and NN3 are quickly localized near $x = 0.4$ and $x = 0.9$, respectively, as shown in Fig. 5 (c). Further refined sampling leads to the accurate localization of $s^{\{1\}}$ and $s^{\{2\}}$.

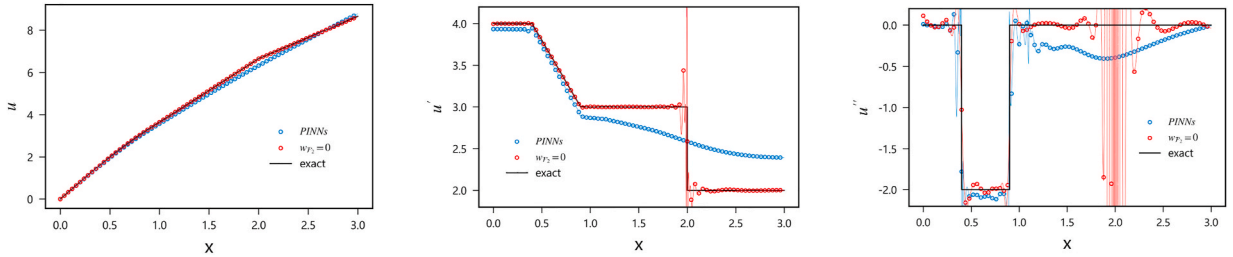
In Fig. 6(a–c), C^0, C^1 and C^2 discontinuity all exhibit sharp gradients. $\mathbb{H}^{\{2\}}(x)$ and $\mathbb{H}^{\{3\}}(x)$ both satisfy the condition of being strictly zero outside the domain and at discontinuity, effectively suppressing the impact of derivative singularities on the network. Additionally, within the defined subdomain $\Omega_i^{\{j\}}$, the function values gradually rise as they move away from discontinuities, allowing the neural network to first fit smooth regions and then gradually approach discontinuities with singularity. This represents an optimal solution approach.

However, the accuracy of AS-PINNs with $\mathbb{H}^{\{3\}}(x)$ is unsatisfactory. Although $\mathbb{H}^{\{3\}}(x)$ exhibits higher continuity, it results in overly smooth transitions between subdomains, which prevents precise localization of discontinuities. Despite minor deviations, incorrect domain decomposition create singularities at discontinuities, which introduce a substantial number of low-information residuals into the loss function, resulting in longer training times and reduced accuracy. Therefore, subsequent experiments will utilize $\mathbb{H}^{\{2\}}(x)$ as the residual activation function.

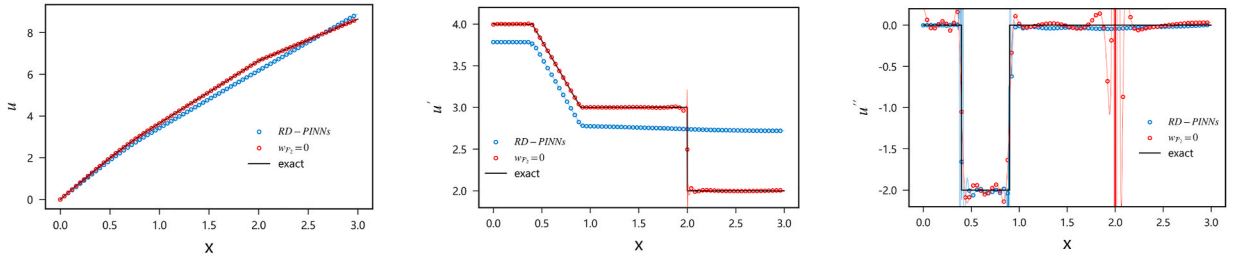
RD-PINNs and XPINNs, which are used to solve discontinuous problems, are compared with AS-PINNs. Although DR-PINNs perform well on high-order differential equation problems, they do not employ a spatial domain decomposition strategy, which prevents them from avoiding the singularities in the solution, resulting in lower solution accuracy. XPINNs can also solve this problem, obtaining high accuracy for u and u'. However, it is important to note that this comparison is not entirely fair because AS-PINNs require optimization of transition points during training to achieve adaptive domain decomposition. In contrast, XPINNs have predefined subdomain that do not need further refinement during training. Despite this, AS-PINNs still achieve accuracy comparable to XPINNs. For a fair comparison, transition points are fixed, as



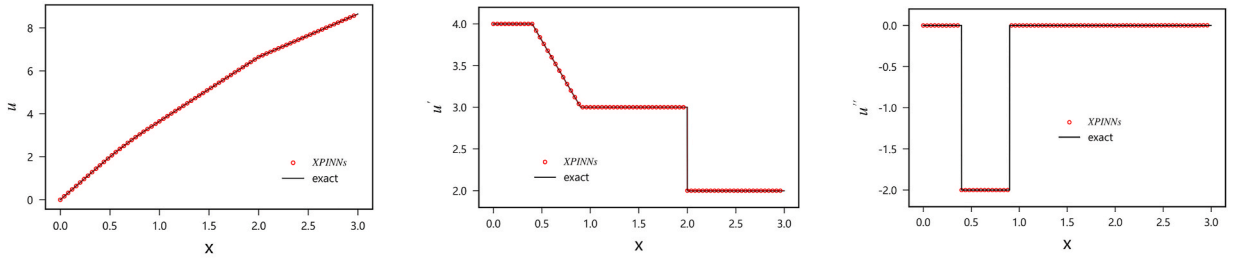
(a, b, c) As-PINNs with different residual activation function to automatically locate and capture discontinuities



(d, e, f) PINNs



(g, h, i) RD-PINNs



(j, k, l) XPINNs

Fig. 6. Comparison between AS-PINNs, PINNs, RD-PINNs, and XPINNs. (First column) Target function. (Second column) First derivative. (Third column) Second derivative.

indicated in Table 1. Even though AS-PINNs require more variables output from the same network structure compared to XPINNs, the results show higher accuracy, with L_2 error in u and u' reaching 10^{-6}, significantly outperforming XPINNs by an order of magnitude. It is believed that the network interface conditions of AS-PINNs use hard constraints, which not only reduce the complexity of the loss function but also naturally enhance network accuracy.

To further clarify the interference of various order derivative fitting tasks in discontinuity problems, set $w_{\mathcal{F}_2}=0$ in PINNs and RD-PINNs. This means that only u' is input, simulating the scenario where the residual of the first-order derivative information dominates. With less input information, v_0 and v_1 actually obtained better results. This indicates that different loss function terms conflict with each other to some extent. When using a network with a smooth activation function to fit a discontinuous function, significant oscillations occur in the primary variable $\widehat{v}_1$ and its derivative $d\widehat{v}_1/dx$.

Similar to interpolation methods, achieving high accuracy at discontinuities requires densely clustered interpolation nodes. Such densely clustered nodes often induce oscillations in the interpolation function. Despite these oscillations, high-order interpolation provides a more accurate representation of rapid changes near discontinuities. For integration operations, results are not significantly affected by local errors. Consequently, due to the accurate interpolation of most values at discontinuities, the impact of these oscillations on low-order variables is minimal. As shown in Fig. 4 (f) and (i) at $x=2$, $\widehat{v}_1$ experiences significant oscillations, whereas $\widehat{v}_0$ maintains relatively high accuracy. However, steep gradients lead to substantial residuals in u''. To balance the residuals of u'' at $x=2$, a regularizing effect is applied to the

Table 1
Results of different Methods for function fitting.

Method	Target Function [a]	Residual activation function	NN [b]	L_2 Relative Error			Self-Adaptive Partition		
				u	u'	u''	$s^{\{1\}}$	$s^{\{2\}}$	$s^{\{3\}}$
ASPINNs	u', u''	$\mathbb{H}^{\{1\}}$	4+4	3.93E-02	1.32E-01	2.87E-01	0.070	0.188	1.040
ASPINNs	u', u''	$\mathbb{H}^{\{2\}}$	4+4	2.06E-05	2.10E-04	9.78E-04	0.400	0.900	2.000
ASPINNs	u', u''	$\mathbb{H}^{\{2\}}$	4+4	6.84E-05	3.26E-04	1.41E-03	0.400	0.900	2.000
ASPINNs	u', u''		4+4	**2.18E-06**	**4.35E-06**	**1.10E-04**		Fixed	
ASPINNs	u', u''	$\mathbb{H}^{\{3\}}$	4+4	1.25E-03	1.94E-02	1.42E-01	0.394	0.904	1.990
PINNs	u', u''		1+0	2.62E-02	1.04E-01	4.34E-01			
PINNs	u'		1+0	4.47E-04	3.37E-02	5.10E+01			
RDPINNs	u', u''		1+1	4.65E-02	1.54E-01	6.05E-01			
RDPINNs	u'		1+1	1.41E-04	4.89E-03	1.95E+01			
XPINNs	u', u''		4+0	5.13E-05	5.44E-05	6.82E-04			

[a] Information of the target function incorporated within the loss function.
[b] Number of $\hat{v}_0^{\{j\}}$ and $\hat{v}_1^{\{j\}}$, such as $\hat{v}_0^{\{0\}}, \hat{v}_0^{\{1\}}, \hat{v}_0^{\{2\}}, \hat{v}_0^{\{3\}}$ and $\hat{v}_1^{\{0\}}, \hat{v}_1^{\{1\}}, \hat{v}_1^{\{2\}}, \hat{v}_1^{\{3\}}$ denoted as "4 + 4".

derivative of $\hat{v}_1$. This results in the $\hat{v}_1$ and $\hat{v}_0$ exhibiting smooth characteristics at the discontinuity location. Whether the final network output shows oscillations or smoothing depends on the relative magnitude of corresponding loss term. The challenge of solving high-order partial differential equations within the PINNs framework, in addition to the instability of high-order derivatives, stems from the mutual influence and balance of derivatives of different orders.

To simplify the presentation, a neural network with three hidden layers and 50 neurons per layer is denoted as 3×50. Experiments were conducted using network structures 3×20, 3×80, 3×110, and 5×50 to investigate the impact of network configurations on the results. The input to the model includes u' and u''. The relative errors are summarized in Table 2. Within a certain range, fitting accuracy improves as the number of network parameters increases. When using the 3×20 network structure, the positioning of $s^{\{1\}}$ exhibited deviations. Although the accuracy of u and u' slightly improved, the accuracy of u'' significantly decreased. When the network width increased to 80, the accuracy of all variables improved, indicating that the enhanced network expressiveness contributed to the observed gains. However, further increasing the network parameters did not result in additional accuracy improvements, likely because excessive parameters impose a training burden and lack sufficient training points to constrain the over-parameterized network. It is worth noting that the examples not undergo extensive hyperparameter tuning. The focus of this work is to identify the challenges PINNs face when solving high-order differential equations with discontinuities and to provide an effective solution. In the subsequent examples, a 3×50 neural network will consistently be used to output all variables.

4. Numerical examples

The Euler beam problem was chosen to validate AS-PINNs due to its inherent complexity involving high-order differential equations. This equation provides a rigorous benchmark for testing capabilities of AS-PINNs, particularly in handling discontinuities such as abrupt changes in material properties or external loads, which pose significant challenges for PINNs. In addition, the results of AS-PINNs solving the sixth-order differential equation are provided in Appendix A.

4.1. Discontinuity caused by force

Fig. 7 illustrates an Euler-Bernoulli beam with discontinuities as considered by RD-PINNs (Luong et al., 2023). The beam is fixed at the left end and subjected to a vertical displacement constraint at a position 1/4 m from the right end. The length is $L = 1$ m, with a cross-sectional width $b = 12$ mm, height $h = 50$ mm, and material elastic modulus $EI = 8 \times 10^4$ MPa. The system has two discontinuous points, i.e., $d_1 = 1/3$, $d_2 = 1/4$. The governing equation of the beam is

$$-\frac{d^2}{dx^2}\left(EI(x)\frac{d^2w(x)}{dx^2}\right) + q(x) = 0 \tag{28}$$

Table 2
Function fitting and self-adaptive domain decomposition results under networks of different sizes.

Method	Residual activation function	NN Structure	L_2 Relative Error			Self-Adaptive Partition		
			u	u'	u''	$s^{\{1\}}$	$s^{\{2\}}$	$s^{\{3\}}$
ASPINNs	$\mathbb{H}^{\{2\}}$	3×20	**2.12E-05**	**1.04E-04**	4.47E-02	0.399	0.900	2.000
ASPINNs	$\mathbb{H}^{\{2\}}$	3×50	6.84E-05	3.26E-04	**1.41E-03**	0.400	0.900	2.000
ASPINNs	$\mathbb{H}^{\{2\}}$	3×80	2.46E-05	1.70E-04	7.40E-04	0.400	0.900	2.000
ASPINNs	$\mathbb{H}^{\{2\}}$	3×110	4.37E-05	2.56E-04	2.48E-03	0.400	0.900	2.000
ASPINNs	$\mathbb{H}^{\{2\}}$	5×50	2.61E-05	8.98E-05	1.47E-03	0.400	0.900	2.000
ASPINNs	$\mathbb{H}^{\{2\}}$	5×80	1.57E-05	1.65E-04	1.88E-03	0.400	0.900	2.000

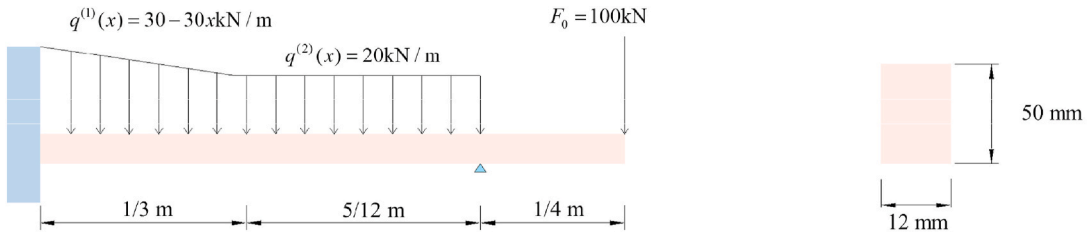


Fig. 7. Euler–Bernoulli beam with discontinuity caused by force.

The relationships among transverse displacement w, rotation φ, curvature $d\varphi/dx$, bending moment M, shear force V, and spatially distributed load q are given by

$$\varphi = -\frac{dw}{dx} \tag{29}$$

$$M = EI\frac{d\varphi}{dx} \tag{30}$$

$$V = \frac{dM}{dx} \tag{31}$$

$$q = -\frac{dV}{dx} \tag{32}$$

This problem exhibits unique characteristics: the function w and its first derivative dw/dx display continuous and smooth features, while the second derivative d^2w/dx^2, the third derivative d^3w/dx^3, and the forth derivative d^4w/dx^4 exhibit discontinuities with varying characteristics. When using a single neural network for solving this problem, it is challenging to accurately capture the discontinuities in high-order derivatives while adequately representing the continuity in lower-order derivatives. For the sake of simplicity and clarity, let rotation φ be denoted as w_1, the curvature $d\varphi/dx$ as dw_1/dx, the bending moment M as w_2, the shear force V as w_3, and the distributed load q as w_4.

Assuming the positions and number of discontinuities in the system are unknown. A fully connected neural network with 3 hidden layers and 50 neurons per layer is used to output the required physical quantities. The Adam optimizer is employed for optimization, and the training is conducted for 100,000 epochs. The weights of all loss terms are initially set to 1. The learning rate is set to 10^{-3} for the first 10,000 epochs, and then reduced to 10^{-4}. 200 residual points are randomly sampled in the domain Ω. Every 10,000 epochs, 20 sampling points are added using the Residual-based Adaptive Refinement (RAR) method, and the weights are updated based on Equation (25). After 40,000 epochs, the weights are no longer updated. The iterative process for the transition point location and the loss function are shown in Fig. 8. The process of self-adaptive domain decomposition, where subnetworks compete through residuals, along with further details, is shown in Fig. 9. Fig. 10 presents the results of AS-PINNs under different network architectures. To enhance the clarity of Fig. 10, only a subset of the results is presented. Table 3 provides the corresponding L_2 relative errors. In the table, the optimal results are underlined, while the second-best results are highlighted in bold.

A concentrated load causes discontinuity in high-order derivatives. Therefore, to ensure the continuity, w and w_1 can be represented by subnetwork $\widehat{w}$ and $\widehat{w}_1$ without causing singularities. w_2 and w_3 are

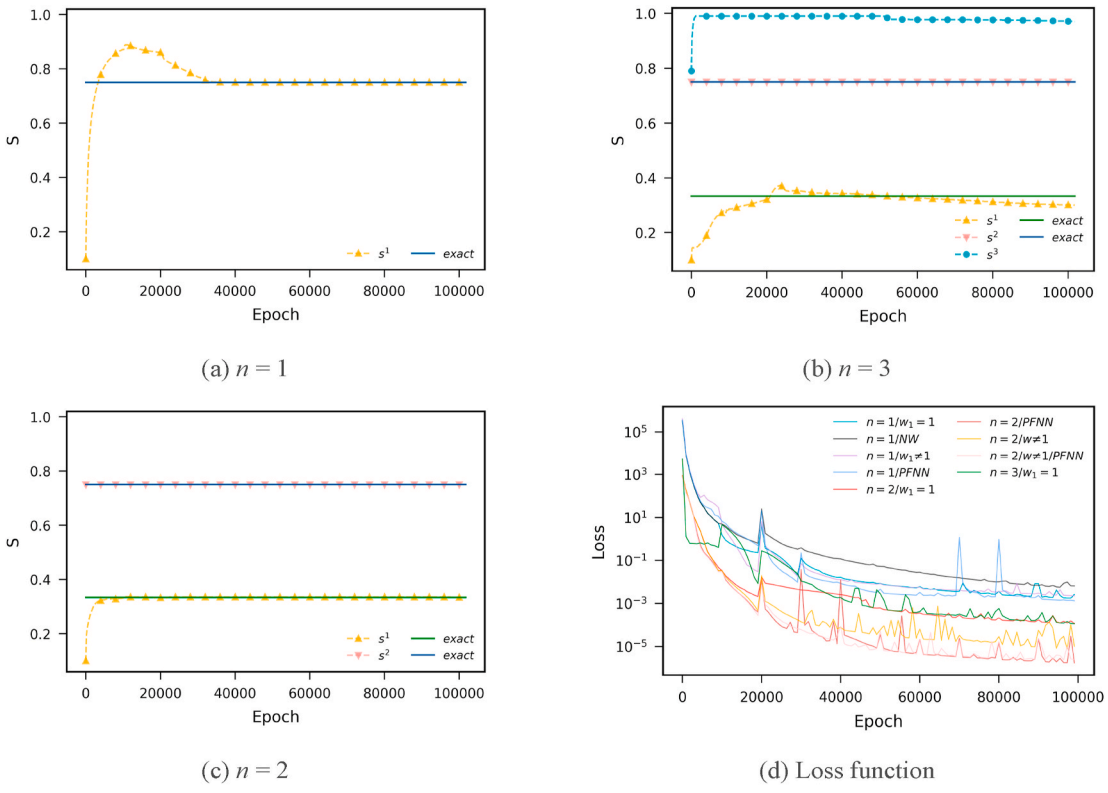


Fig. 8. (a–c) The convergence of interface location of AS-PINNs and (d) loss function.

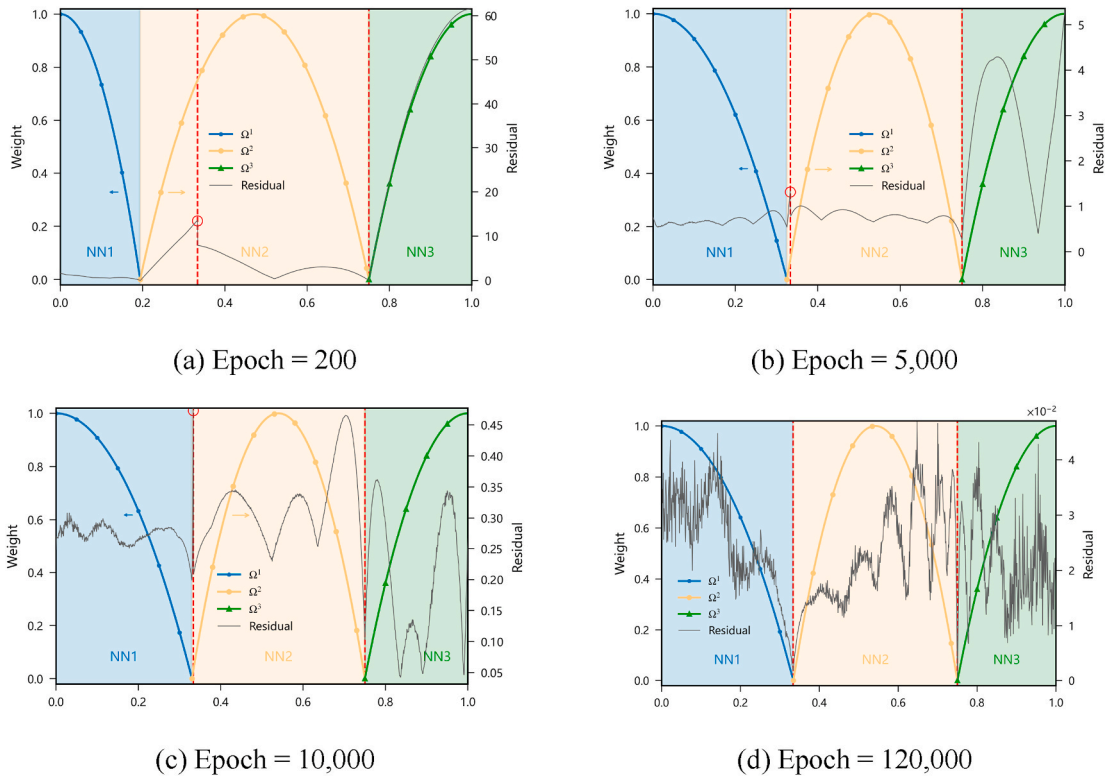


Fig. 9. Self-Adaptive domain decomposition with discontinuity caused by force. The results shown correspond to $n = 2$, where $s^{\{1\}}$ is trainable, and $s^{\{2\}}$ is fixed.

represented by multiple subnetworks $\widehat{w}_2^{\{j\}}$ and $\widehat{w}_3^{\{j\}}$. Initialize $n = 1$, the model is denoted as "$n = 1/w_1 = 1$". The structure of the network output is "1 + 1+2 + 2", indicating that the number of sub-NNs $\widehat{w}^{\{j\}}$, $\widehat{w}_1^{\{j\}}, \widehat{w}_2^{\{j\}}, \widehat{w}_3^{\{j\}}$ is 1, 1, 2 and 2. Setting the transition set $\mathbf{S} = \{s^{\{1\}}\}$, the computational domain is divided into two regions $\Omega^{\{1\}} = [0, s^{\{1\}})$ and $\Omega^{\{2\}} = [s^{\{1\}}, L]$. Networks $\widehat{w}^{\{j\}}$ and $\widehat{w}_1^{\{1\}}$, are set in Ω to represent w and w_1. Networks $\widehat{w}_2^{\{1\}}$, $\widehat{w}_2^{\{2\}}$ and $\widehat{w}_3^{\{1\}}$, $\widehat{w}_3^{\{2\}}$ are set in $\Omega^{\{1\}}$ and $\Omega^{\{2\}}$ to represent w_2 and w_3.

Discontinuities are distributed across physical quantities of different orders. Assigning higher weights to the physical quantities where discontinuities occur can accelerate the self-adaptive domain decomposition process. The initial rapid weight updates enable AS-PINNs to swiftly identify regions near discontinuities, as shown in Fig. 8. Once the transition set $\mathbf{S}$ stabilized, weight updates are ceased to minimize computational cost. Residual points sampled by RAR further improved the resolution at discontinuities, enhancing overall result accuracy. $\mathbb{H}_{\Omega^{\{j\}}}$ is not utilized during RAR sampling, as its zero value at discontinuities limited the sampling of residual points in these regions. Since this solving method does not involve computing the derivative of $q(x)$, the discontinuity at d_1 does not cause singularities, resulting in a relatively weak discontinuity. As the training proceeds, the transition point $s^{\{1\}}$ tends to stabilize near d_2. With further refinement of the residual points, $s^{\{1\}}$ is precisely determined to be at d_2. The accuracy of all physical quantities has reached 10^{-3}, proving that AS-PINNs have strong capabilities in capturing discontinuities and adaptive partitioning when solving differential equations.

Table 4 outlines the process of adjusting weights for the loss function items of the "$n = 1/w_1 = 1$" model. Table 3 presents the results obtained without applying this method. Since the discontinuities in this example are concentrated in a single physical quantity and are unique, $s^{\{1\}}$ is accurately determined at d_2 even without using adaptive weight adjustment strategy. But the L_2 relative error of each variable only reached 10^{-2} where only Q achieving 10^{-3}, suggesting that $w_{\mathscr{F}_4}$ is too large. Inversely, in the model using the adaptive weight adjustment strategy, $w_{\mathscr{F}_4}$ is significantly reduced while $w_{\mathscr{F}_1}$ remains maximal, aligning with the relatively small magnitude of w_1. Adjusting the weights ensures that each task has a similar magnitude, effectively avoiding the imbalance of loss function terms. This outcome demonstrates the effectiveness of the adaptive weight adjustment strategy.

The method of using multiple subnetworks to represent variables with discontinuities and using single network to represent variables without discontinuities has proven successful. However, w and w_1 are not rigorous continuous functions and exhibit c^3 and c^2 discontinuities, respectively. When using continuous networks $\widehat{w}$ and $\widehat{w}_1$ to represent and convey gradient information, errors are inevitable, especially at the intersections of single subnetwork and multiple subnetworks, i.e., $\widehat{w}_1$ and $\widehat{w}_2^{\{j\}}$. The results shown in Fig. 10 (e) (f) indicate that errors are concentrated around d_2 in the "$n = 1/w_1 = 1$" model. Consequently, the function graph of $d\widehat{w}_1/dx$ exhibits smooth characteristics. Although w_1 has c^2 discontinuity, $\widehat{w}_1$ is a continuous function theoretically. Despite the RAR method focusing on sampling at d_2, it fails to eliminate inherent defects, demonstrating the importance of framework selection to specific problems for accuracy and efficiency. Here, $\widehat{w}_2^{\{j\}}$ shows high precision instead of smoothing effects, as its output is influenced not only by $d\widehat{w}_1/dx$ but also by $\widehat{w}_3$. The final precision output of $\widehat{w}_2^{\{j\}}$ depends on the relative magnitude of different loss terms.

Two methods can be used to improve the gradient flattening and low precision issues caused by a single neural network approximating high-order discontinuous variable. Firstly, multiple neural networks $\widehat{w}_1^{\{j\}}$ can be used to approximate $\widehat{w}_1$, denoted as "$n = 1/w_1 \neq 1$". The structure of the network output is "1 + 2+2 + 2". Due to the relatively high smoothness of $\widehat{w}_1^{\{j\}}$ and the absence of significant discontinuities, the positioning of $s^{\{j\}}$ does not depend on $\widehat{w}$. Replacing $\mathbb{H}_{\Omega^{\{j\}}}$ with $\mathbb{G}_{\Omega^{\{j\}}}$ can reduce computational costs. As shown in Fig. 10 (e), the flattening effect of $d\widehat{w}_1/dx$ is alleviated, and the curve exhibits sharp gradient. From the corresponding error curve Fig. 10 (f), it is observed that the high error at

position d_2 is eliminated. However, the L_2 relative error does not decrease. It is noteworthy that due to the convergence of multiple networks and a single network from w_1 and w_2 to w_0 and w_1, the concentration of the error at d_2 from w_2 shifts to w, as shown in Fig. 10 (f) and (b). Using multiple networks to approximate high-order discontinuous physical fields can alleviate the local gradient flattening effect of $d\widehat{w}_1/dx$ but does not significantly improve overall accuracy and may even degrade it. This is related to optimizer and the distribution of the error. This indicates that, when minimizing the number of networks while

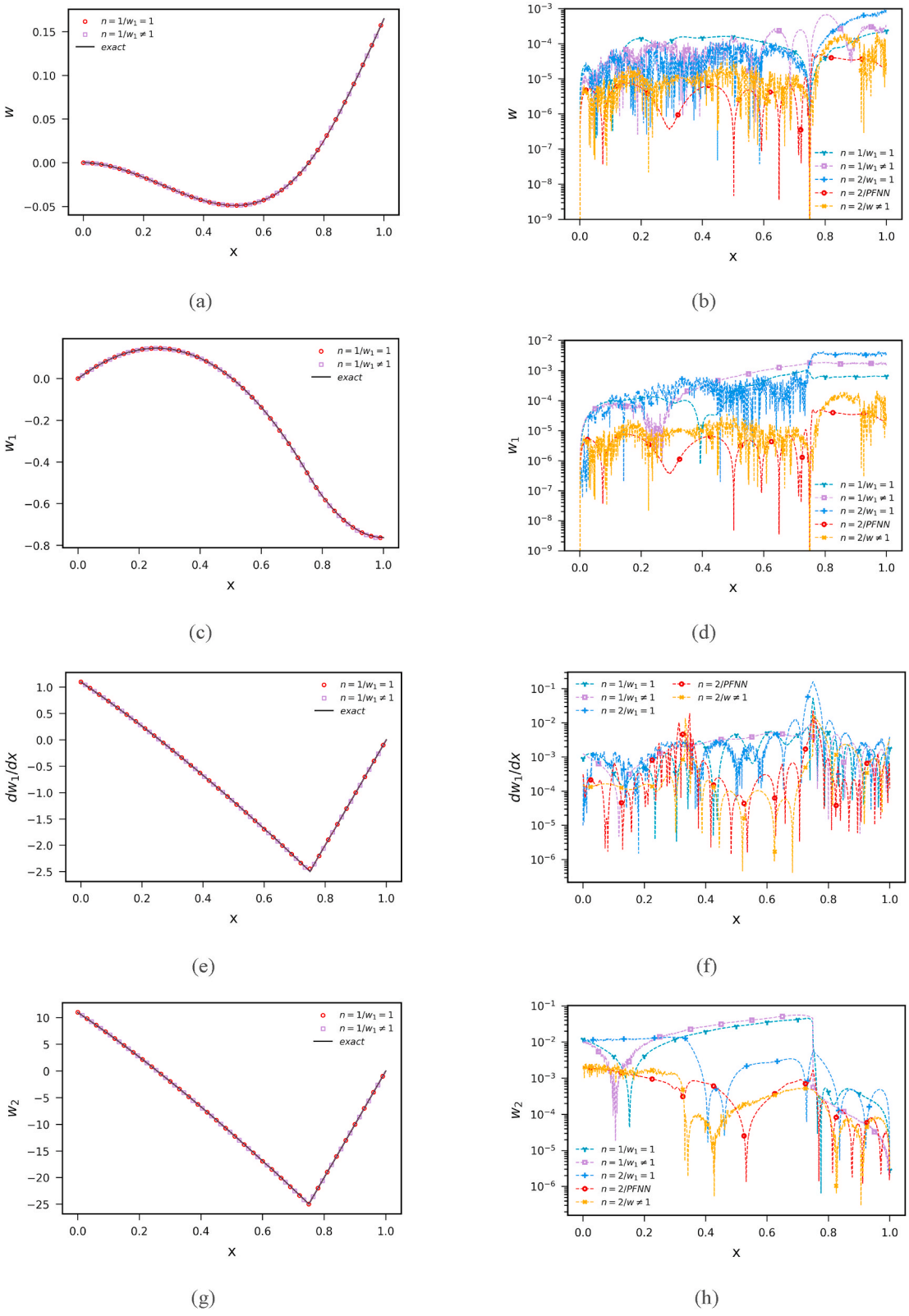


Fig. 10. (First column) Results of different physical variations of the of the beam with discontinuity caused by force, and (Second column) point-wise errors. (a–b) Displacement, (c–d) Slope, (e–f) Curvature, (g–h) Bending moment, (i–j) Shear force, (k–l) Distributed force.

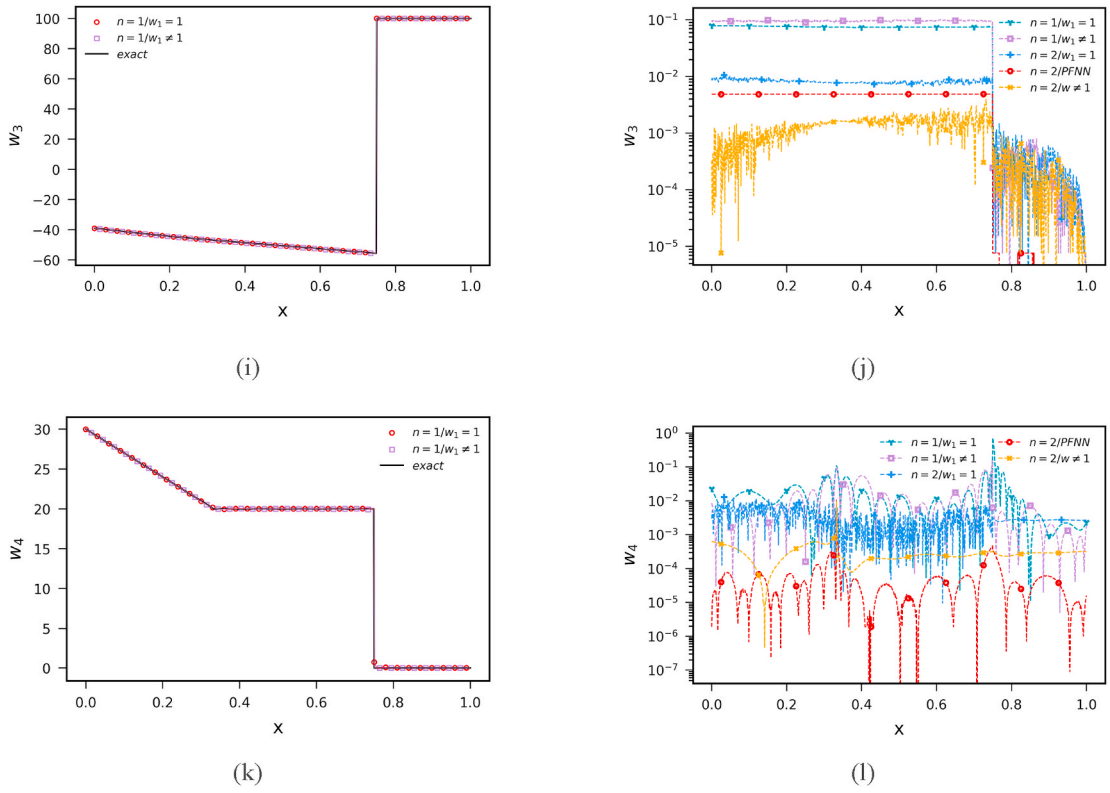


Fig. 10. (*continued*).

balancing accuracy and computational efficiency, using multiple networks to approximate the C^0 or C^1 discontinuous variables represents a relatively optimal approach.

Another approach is to customize networks based on the characteristics of each order of physical quantities. Specifically, for the physical quantities exhibit high-order discontinuities (e.g., $\widehat{w}_1$ and $\widehat{w}_2$), activation functions with steep gradient characteristics should be used, such as *Elu*. This structure is generally referred to as the "Parallel Fully-Connected Network", denoted as "PFNN". Table 3 presents the results, with the labels in the "NN" column indicating "1 + 1+2 + 2" and the labels in the "NN" column indicating "PFNN". While this network framework does not show significant advantages when $n = 1$, it exhibits relatively high accuracy in subsequent experiments. It is important to note that the setup and configuration are not adjusted to optimize model performance but to explain the limited accuracy from a mathematical perspective and provide corresponding solutions.

When $n = 1$, $s^{\{1\}}$ is optimally positioned at d_2. To locate other potential discontinuities and improve search efficiency, $s^{\{2\}}$ and $s^{\{3\}}$ are added. $s^{\{2\}}$ is fixed at 0.75, $s^{\{1\}}$ and $s^{\{3\}}$ are set at the left and right ends to search for discontinuities. Based on previous conclusions, using a single network to approximate w and w_1, and using multiple networks to approximate w_2 and w_3, similar training strategies are applied. The structure of the network output is "1 + 1+4 + 4" and "FNN". $s^{\{1\}}$ will be positioned near d_1, while $s^{\{3\}}$ will quickly be squeezed out of the domain, consistent with the absence of discontinuities in the right end domain. It is believed that the weights at the transition points are set to zero. Introducing excessive transition points in regions without discontinuities may render the original problem ill-posed, thereby increasing its complexity and difficulty. To minimize the loss function, redundant transition points are squeezed out of the domain. Here, the range of $s^{\{3\}}$ is forced to be fixed at (d_2,0.95m], denoted as "$n = 3/w_1 = 1$", providing the corresponding results. The iterative process for the transition point location are shown in Fig. 8 (b).

Based on $n = 3$, the performance of $s^{\{1\}}$ and $s^{\{3\}}$ confirms that there is only one discontinuity within the interval [0,0.75). Considering that this interval contains only one discontinuity, $n = 2$ is set, with transition points $s^{\{1\}}$ and $s^{\{2\}}$. For $q(x)$, a method of gradient enhancement (Yu et al., 2022) is used, i.e., for discontinuous C^1, to reveal its peculiar singularity, allowing the transition point to better capture the position of the discontinuity d_1. Although the method of gradient enhancement is applied to find the discontinuity of C^1, AS-PINNs do not need to adjust the solving strategy based on the characteristics of the discontinuity. When $n = 1$, AS-PINNs have already accomplished the solution well. This is only to illustrate how to capture discontinuities through the variation of n and complete the domain decomposition and framework adjustment. Here, four sets of experiments are set, respectively denoted as "$n = 2/w_1 = 1$", "$n = 2/PFNN$", "$n = 2/w \neq 1$", and "$n = 2/w \neq 1/PFNN$", where "$w \neq 1$" indicates using multiple networks to approximate w. The numbers of the output physical quantities $\widehat{w}^{\{j\}}$, $\widehat{w}_1^{\{j\}}$, $\widehat{w}_2^{\{j\}}$ and $\widehat{w}_3^{\{j\}}$ are respectively "1 + 1+3 + 3," "1 + 1+3 + 3," "3 + 3+3 + 3," "3 + 3+3 + 3; " the adopted network structures are "FNN," "PFNN, " "FNN," "PFNN."

The iterative process for the transition point location of "$n = 2/w_1 = 1$" are shown in Fig. 8 (c). Fig. 9 provides more information on how the subnetworks compete based on the size of the residuals, thereby completing the self-adaptive domain decomposition. In the early stages of training, the subdomain of NN2 exhibits larger residuals, which allows NN2 to prevail in the competition with NN1, causing the interface between NN2 and NN1 to shift to the right. Through further sampling, $s^{\{1\}}$ is accurately positioned at $x = 0.33$. Fig. 10 shows the point-wise errors of some experiments, and Table 3 shows the L_2 relative errors. Theoretically, $n = 2$ matches the actual number of discontinuities in the system, and AS-PINNs should have optimal performance. These three sets of experiments demonstrated very high accuracy, especially "$n = 2/PFNN$" and "$n = 2/w \neq 1$", with L_2 relative errors averaging 10^{-4}, and the accuracy of some physical quantities reaching 10^{-5}. The PFNN network structure did not achieve good results when $n = 1$, possibly

Table 3
Results of different Methods for discontinuous beam problem caused by force.

NN [a]	NN [b]	L_2 Relative Error					Transition Points		
		w	$w_1(\varphi)$	$w_2(M)$	$w_3(V)$	$w_4(Q)$	$s^{\{1\}}$	$s^{\{2\}}$	$s^{\{3\}}$
1+1+2+2	FNN	2.21E-3	1.23E-3	1.64E-3	1.00E-3	2.22E-3		0.75	
1+1+2+2 [c]	FNN	7.00E-2	8.75E-2	4.07E-2	1.72E-2	2.15E-3		0.75	
1+2+2+2	FNN	3.76E-3	2.98E-3	2.16E-3	1.27E-3	1.12E-3		0.75	
1+2+2+2	PFNN	6.21E-3	5.72E-3	3.10E-3	6.15E-4	1.76E-3		0.75	
1+1+4+4	FNN	3.65E-3	4.80E-3	7.56E-4	4.65E-4	1.46E-3	0.32	0.75	0.89
1+1+3+3	FNN	4.69E-3	5.02E-3	5.87E-4	1.09E-4	7.05E-5	0.34	0.75	
1+1+3+3	PFNN	**3.55E-4**	**7.61E-5**	6.77E-5	6.48E-5	1.84E-5	0.33	0.75	
3+3+3+3	FNN	9.29E-4	5.52E-4	7.32E-5	**1.96E-5**	2.30E-5	0.33	0.75	
3+3+3+3	PFNN	2.57E-4	7.28E-5	**3.40E-5**	2.78E-5	**1.79E-5**	0.33	0.75	
3+3+3+3	FNN	5.67E-4	1.73E-4	3.38E-5	1.38E-5	1.64E-5		Fixed	
DR-PINNs		3.08E-3	4.80E-3	1.48E-2	1.83E-1			Fixed	
XPINNs		2.54E-3	1.04E-3	1.14E-3	6.91E-4	7.34E-3		Fixed	

[a] Number of outputs $\widehat{w}^{\{j\}}, \widehat{w}_1^{\{j\}}, \widehat{w}_2^{\{j\}}, \widehat{w}_3^{\{j\}}$.
[b] Network structure used, "FNN" is a fully connected network, "PFNN" is a parallel fully connected network with a special activation function.
[c] Adaptive weights are not used.

Table 4
The iterative process of weights for loss function terms.

Epoch	$w_{\mathcal{F}_1}$	$w_{\mathcal{F}_2}$	$w_{\mathcal{F}_3}$	$w_{\mathcal{F}_4}$
0	1	1	1	1
1w	1.00E+00	7.75E-01	8.04E-01	1.67E-01
2w	1.00E+00	5.88E-02	4.24E-02	9.02E-02
3w	1.00E+00	1.59E-01	2.15E-01	7.44E-01

because the single network could not adequately complete the approximation of the c^2 discontinuity at d_1, causing the RAR algorithm to focus its sampling here, ignoring the overall accuracy. When $n = 2$, the drawback of using a single network to output the discontinuous function is completely avoided. This releases a significant amount of training effort in AS-PINNs, allowing them to train while considering global accuracy. The experiment with "$n = 2/w \neq 1$" approximation of the base function in a single region, avoiding the use of a single network to fit the high-order discontinuous function, while also achieving a significant improvement in accuracy.

The loss function graphs indicate that the "$n = 2/\ w \neq 1/\ PFNN$" model and the "$n = 2/PFNN$" model have the smallest loss functions, followed by the "$n = 2/w \neq 1$" model. Under different settings, AS-PINNs have demonstrated relatively high accuracy, surpassing the RD-PINNs results in the literature. AS-PINNs have achieved multiple orders of magnitude improvements, particularly for high-order physical quantities. In practical applications, using multiple networks to approximate the physical quantities with C^0 and C^1 discontinuities, and using a single network to approximate high-order continuous physical quantities within the computational framework, has already met the needs of most scenarios. This fully illustrates the concept of AS-PINNs, aiming to achieve a framework of highly efficient and affordable computing overhead tailored to the characteristics of the problem.

In addition, the solving results of DR-PINNs from the literature (Luong et al., 2023) are used for comparison. XPINNs are also set up and applied to solve this case. The corresponding L_2 relative errors are presented in Table 2. According to the literature, DR-PINNs demonstrate a significant advantage over PINNs when solving smooth high-order problems; however, when the system contains discontinuities, singularities in differentiation lead to a severe decrease in the accuracy of high-order physical quantities. XPINNs achieve an accuracy of 10^{-4} for w_3 but only show an accuracy of 10^{-3} for other physical quantities. AS-PINNs not only successfully complete the automatic adjustment of the computational domain and the network framework, but also exhibit superior accuracy compared to XPINNs. Nearly all physical quantities achieve or approach an accuracy of 10^{-5}. Furthermore, after fixing the transition points of AS-PINNs, the accuracy of certain physical quantities improves further, fully demonstrating the competitiveness of AS-PINNs.

4.2. Discontinuity caused by force and material

Consider the beam depicted in Fig. 11 to evaluate the capability of AS-PINNs in capturing discontinuities and performing adaptive domain decomposition for multiple physical quantities of different orders experiencing discontinuities. The cross section of the beam is the same as the one in Fig. 10. But in the range of [0–0.5) m, the elastic modulus of the material is $EI_1 = 8 \times 10^4$ MPa; in the range of [0.5–1) m, the elastic

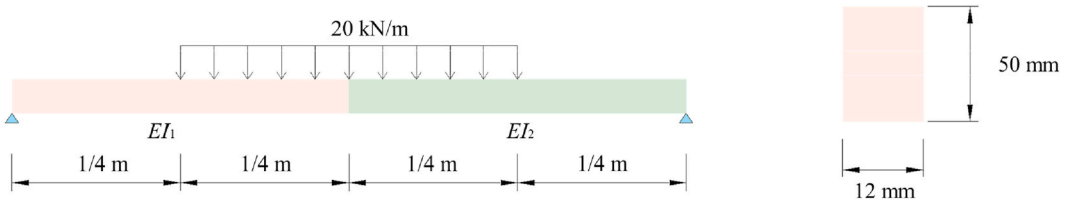


Fig. 11. Euler–Bernoulli beam with discontinuity caused by force and material.

modulus of the material is $EI_2 = 4 \times 10^4$ MPa. The beam is subjected to a partially distributed force $q(x)$ as shown in Fig. 11. In this system, discontinuities occur at $d_1 = 0.25$ m, $d_2 = 0.5$ m, and $d_3 = 0.75$ m, affecting physical quantities of different orders. When using a single neural network to solve this problem, accurately capturing discontinuities in certain derivatives while maintaining continuity in others is challenging.

The case of n variation has been previously discussed and is not addressed here. Consider $n = 3$ with the addition of network $\widehat{EI}^{\{j\}}$. For $n = 3$, the set of transition point is defined as $\mathbf{S} = \{s^{\{1\}}, s^{\{2\}}, s^{\{3\}}\}$, forming the subdomains $\Omega^{\{1\}} = [0, s^{\{1\}})$, $\Omega^{\{2\}} = [s^{\{1\}}, s^{\{2\}})$, $\Omega^{\{3\}} = [s^{\{2\}}, s^{\{3\}})$, $\Omega^{\{4\}} = [s^{\{3\}}, s^{\{4\}}]$. Set subnetworks $\widehat{w}^{\{j\}}$, $\widehat{w}_1^{\{j\}}$, $\widehat{w}_2^{\{j\}}$, $\widehat{w}_3^{\{j\}}$, $\widehat{EI}^{\{j\}}$ to approximate w, w_1, w_2, w_3, EI within $\Omega^{\{j\}}$, where $j = 1, 2, 3, 4$. The derivative operator can reveal the singularity, allowing the transition points to better capture the location of the discontinuity. Here, for the approximation of EI, the gradient enhancement method is used. The discontinuities of higher-order physical quantities have a limited effect on the original function through multiple integrations. The original function often exhibits smooth characteristics and lacks sufficient features to allow AS-PINNs to effectively perform localization and domain decomposition. Therefore, the operator $\mathbb{G}_{\Omega^{\{j\}}}$ is implemented for $\widehat{w}_1^{\{j\}}$ to enhance the training efficiency.

This section sets up a model to discuss the effect of the RAR algorithm. Within the domain Ω, 200 residual points are randomly sampled. A fully connected neural network with 3 hidden layers and 50 nodes per layer is used to output the required variables. The Adam optimizer is used for optimization, with 140,000 iterations of training. All loss function terms initially have a weight of 1. For the first 60,000 iterations, the learning rate is set to 10^{-3}, and subsequently reduced to 10^{-4}. Every 20,000 iterations, the RAR algorithm is used to add 20 residual points, and Equation (24) is used to update the weight ($w_{\mathscr{F}_6} = 1$). After 60,000 iterations, the RAR algorithm adds 20 residual points at every subsequent 20,000 iterations, but the weights are no longer updated. Similarly, when using RAR sampling, $\mathbb{H}_{\Omega^{\{j\}}}$ is not set for residual computation.

The results of AS-PINNs with RAR (W-RAR) and without RAR (O-RAR) are shown in Fig. 12, Fig. 14, and Table 5. In the table, the optimal results are underlined, while the second-best results are highlighted in bold. More information on how the subnetworks complete the self-adaptive domain decomposition can be found in Fig. 13.

Discontinuities caused by different factors can affect physical quantities at different orders. In this case, the factors causing discontinuities are not singular, resulting in the influence of the discontinuities being distributed across physical quantities of different orders. Improper weight settings or training point collection can cause AS-PINNs to capture only some of the discontinuity points, neglecting others, which

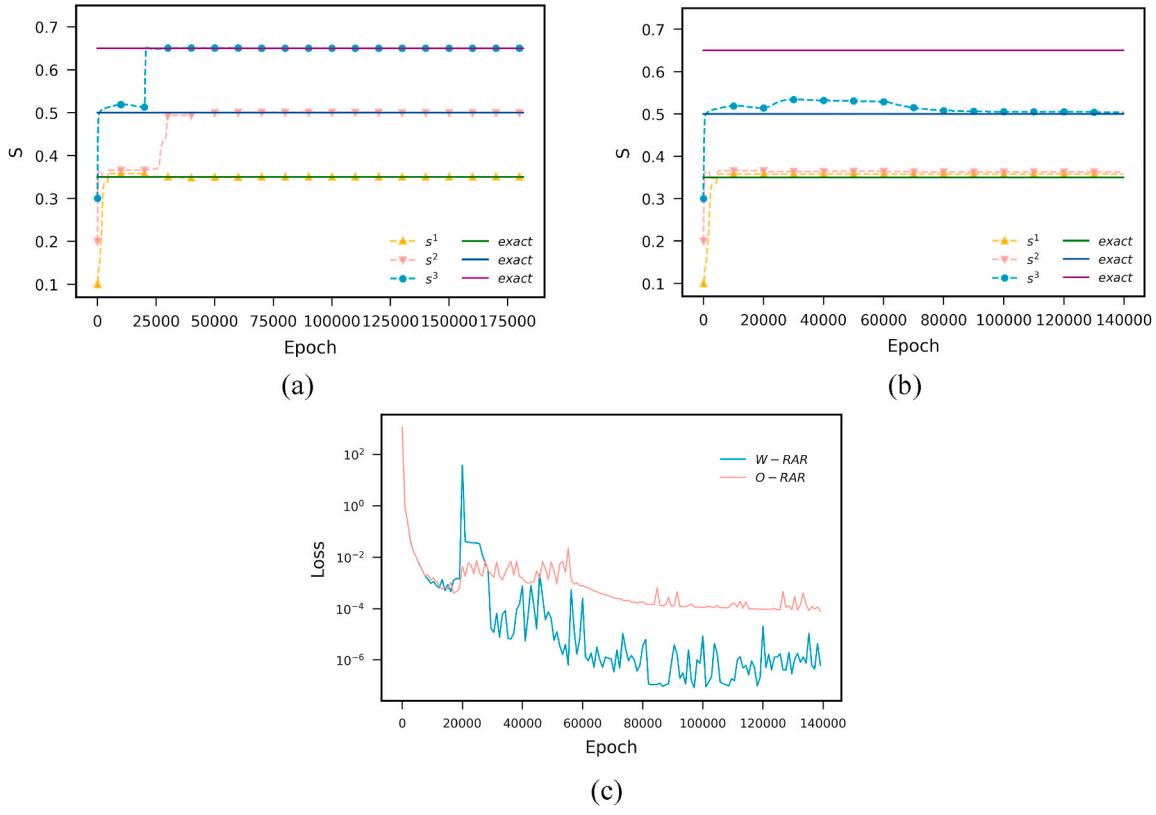


Fig. 12. The convergence of interface location of AS-PINNs (a) with and (b) without RAR, and (c) the loss function.

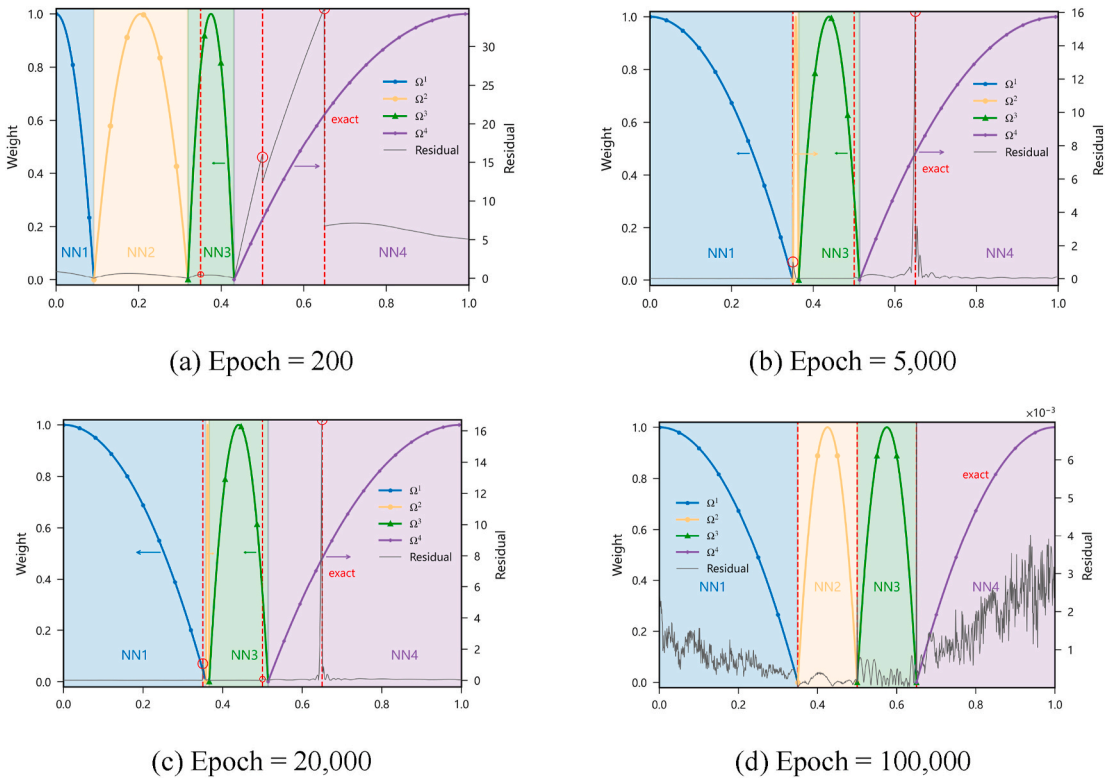


Fig. 13. Self-Adaptive domain decomposition with discontinuities caused by force and material. The results shown are obtained using AS-PINNs with RAR.

leads to failure in solving the problem. This illustrates that when discontinuities are caused by multiple factors, the difficulty of self-adaptive domain decomposition in AS-PINNs increases. During the first 20,000 iterations, $s^{\{1\}}$ and $s^{\{2\}}$ were both assigned near d_1, while $s^{\{3\}}$ was assigned near d_2. AS-PINNs failed to capture the discontinuity at d_3. Unlike the smooth movement of transition points in Fig. 8, the transition points in Fig. 12 stagnated for a long time during the early stages of training, with $s^{\{2\}}$ and $s^{\{3\}}$ failing to update further. This indicates that the residuals provided by the current training points were insufficient to offer enough solution information for subnetworks to update. After 20,000 iterations, new residual points were collected using the RAR algorithm. With more solution information, $s^{\{3\}}$ successfully moved to d_3, and $s^{\{2\}}$ also further moved toward d_2. In contrast, AS-PINNs without the RAR algorithm lacked sufficient solution information to break the current competition balance between the subnetworks. Despite long training, $s^{\{2\}}$ and $s^{\{3\}}$ did not update further. This is also reflected in the loss function curve in Fig. 12. Although there was a significant increase in the loss function curve after applying the RAR algorithm, it quickly decreased again and fell below the AS-PINNs without RAR. This further demonstrates that adjusting the solution mode according to the information from the solution is advantageous.

In Fig. 13, the subnetwork with a larger residual in the subdomain will win the competition. Therefore, during the first 20,000 epochs, the interface between NN4 and NN3 will move to the right. Although the transition points are not accurately located at the positions of d_1 and d_2, their residuals are small, which is related to the weights provided by the residual activation function and the function image learned by the network. For instance, at 200 epochs, the subnetworks have not yet learned the accurate image of w_4, so no significant residual is observed at the d_1 position. After further learning and sampling in subsequent iterations, the residual anomalies at all discontinuity points will be learned by the subnetworks and further optimized.

In Fig. 14, the image of AS-PINNs (W-RAR) shows a high degree of consistency with the actual values, although different physical quantities at various orders exhibit different discontinuity features. Furthermore, all physical quantities directly output by the network, i.e., those connected by hard constraints, did not show large residuals at the interface positions.

Due to the lack of capability in solving discontinuous problems, DR-PINNs and PINNs are not compared here, and the focus is on comparing AS-PINNs with XPINNs, thus demonstrating the competitiveness of AS-PINNs. The relative errors in the L_2 norm for the solutions of AS-PINNs and XPINNs are shown in Table 5. Although without using the RAR algorithm, the solution accuracy of AS-PINNs only reaches 10^{-1}, indicating failure in the solution, this does not imply that AS-PINNs lack competitiveness. On the contrary, with the help of adaptive weights and the RAR algorithm, AS-PINNs exhibit strong robustness, allowing flexible adjustments of weights and the frequency of residual point adjustments to avoid overfitting. The overall solution accuracy for solving multi-factor induced discontinuities reaches 10^{-4}. In contrast, XPINNs have w_1 accuracy of only 10^{-2} and w of 10^{-1}. This is because the known condition in this problem is w_4, and transitioning from w_4 to w requires multiple variables, resulting in cumulative errors. AS-PINNs not only avoid this issue but also achieve further improvement in accuracy when the same conditions as XPINNs are maintained, with predefined subdomains. The overall accuracy reaches 10^{-5}, and w_4 reaches 10^{-6}.

It is important to clarify that the RAR algorithm presented in this context does not aim to directly improve computational accuracy via sampling in PINNs. Instead, it contributes supplementary solving information by promoting adversarial interactions and competition within the network, which assists AS-PINNs in performing self-adaptive domain decomposition.

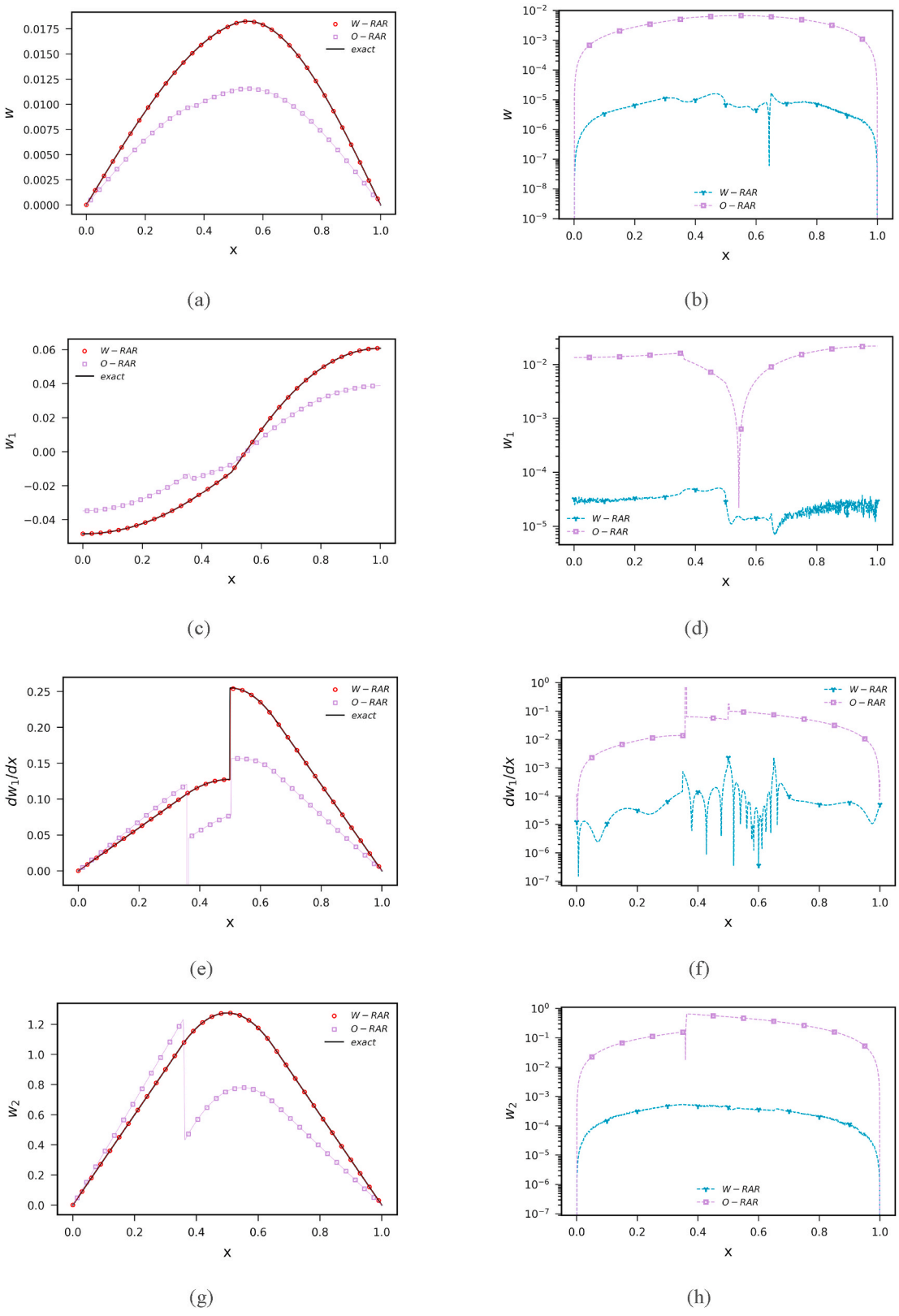


Fig. 14. (First column) Results of different physical variations of the beam with discontinuity caused by force and material, and (Second column) corresponding errors. (a–b) Displacement. (c–d) Slope. (e–f) Curvature. (g–h) Bending moment. (i–j) Shear force. (k–l) Distributed force.

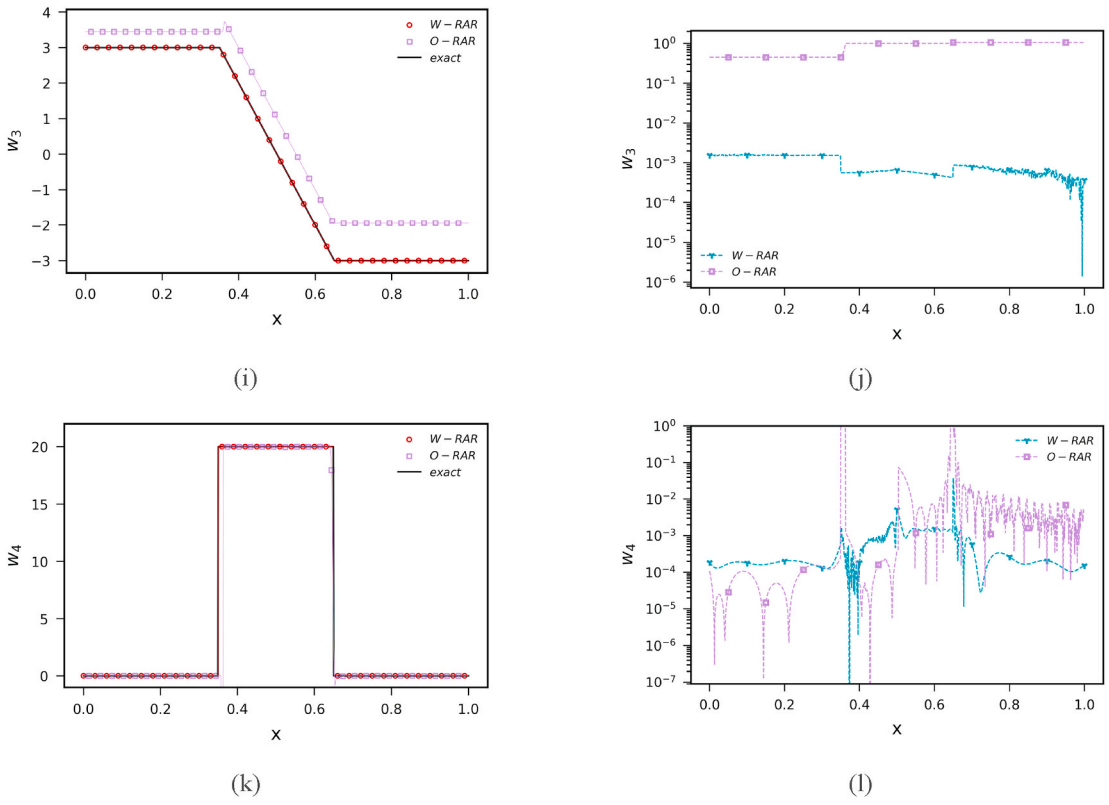


Fig. 14. (*continued*).

Table 5
Results of different Methods for discontinuous beam problem caused by force and material.

NN [a]	RAR	L_2 Relative Error					Transition Points		
		w	$w_1(\varphi)$	$w_2(M)$	$w_3(V)$	$w_4(Q)$	$s^{\{1\}}$	$s^{\{2\}}$	$s^{\{3\}}$
4 + 4+4 + 4	Yes	**6.08E-04**	**7.45E-04**	**3.92E-04**	**3.85E-04**	**1.57E-04**	0.350	0.500	0.650
4 + 4+4 + 4	No	3.56E-01	3.58E-01	3.75E-02	3.23E-01	5.00E-1	0.358	0.363	0.504
4 + 4+4 + 4	Yes	2.66E-05	1.13E-04	1.24E-05	2.96E-05	8.21E-06	Fixed		
XPINNs	Yes	1.36E-02	2.29E-03	6.43E-04	8.38E-04	2.94E-04	Fixed		

[a] Number of outputs $\widehat{w}^{\{j\}}, \widehat{w}_1^{\{j\}}, \widehat{w}_2^{\{j\}}, \widehat{w}_3^{\{j\}}$.

5. Conclusion

Despite the significant success in solving differential equations using neural networks, PINNs have shown poor performance in handling discontinuities. This study demonstrates the singularities induced by discontinuities and the interactions among different physical quantities in high-order differential equations. By utilizing the differences in residuals between subdomains, the residual activation function is proposed to achieve adversarial interactions between subnetworks and the automatic adjustment of the computational domain. To address the challenge of ensuring the continuity condition of moving interface with soft constraints, a hard constraint method for interface conditions is proposed. AS-PINNs eliminate the need for cumbersome operations, including the manual setup of interface conditions, the collection of training points at interface locations, and the inefficient manual decomposition of computational domains. Various PINNs methods were compared in function approximation and numerical problems, highlighting the respective limitations of XPINNs and DR-PINNs within the scope of high-order differential equations with discontinuities. AS-PINNs achieve at least an order of magnitude improvement in accuracy compared to XPINNs and RD-PINNs, with a 2 to 3 orders of magnitude improvement in the accuracy of certain derivatives. The computational speed is improved by a factor of 9 in solving sixth-order differential equations. This paper is expected to provide new insights for expanding the computational framework of PINNs and developing new application scenarios.

However, this paper also has several limitations. The numerical examples in this paper focus on Euler beams, and the adaptive domain decomposition capability of AS-PINNs requires further validation through more diverse test cases. Additionally, the domain decomposition capability of AS-PINNs needs to be extended to irregular computational domains. Future research could investigate the application of

AS-PINNs to additional scenarios, such as defect detection, topology optimization, and high-dimensional problems. A comprehensive study of the theoretical foundations of self-adaptive domain decomposition strategies, along with the optimization of implementation approaches, could further enhance the efficiency of AS-PINNs.

CRediT authorship contribution statement

Mingsheng Peng: Writing – original draft, Visualization, Software, Methodology, Data curation. **Hesheng Tang:** Writing – review & editing, Supervision, Methodology, Investigation, Funding acquisition. **Yingwei Kou:** Writing – review & editing, Validation, Methodology.

Data availability

The code is available at https://github.com/Ning343/AS-PINNs.git.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by National Natural Science Foundation of China (52378184).

A. AS-PINNs for sixth-order differential equation

The study was extended to investigate the performance of AS-PINNs for a sixth-order PDE. Specifically, all variables in the benchmark problem from Section 5.1 were elevated by two orders, transforming the original fourth-order differential equation into a sixth-order one

$$-\frac{d^2}{dx^2}\left(EI(x)\frac{d^2}{dx^2}\left(\frac{d^2}{dx^2}y(x)\right)\right)+q(x)=0 \tag{A.1}$$

and

$$h=\frac{dy}{dx} \tag{A.2}$$

$$w=\frac{dh}{dx} \tag{A.3}$$

Additional boundary conditions, denoted as $y(0)=0$, $h(0)=0$, were introduced, while the remaining settings were kept consistent with Section 5.1. The results are summarized in Table A1. In the table, the optimal results are underlined, while the second-best results are highlighted in bold. The following conclusions were drawn: AS-PINNs maintained high accuracy while completing self-adaptive domain decomposition. XPINNs experienced a significant accuracy decline, regardless of whether RAR was applied. Moreover, XPINNs required 1615 s for training, compared to AS-PINNs, which only needed 564 s in standard settings and 179 s with fixed training points—approximately one-ninth of XPINNs' training time.

These differences arise because XPINNs require additional loss terms for each interface condition, which increase from 7 to 11 when the PDE order rises from 4 to 6. Higher-order interface conditions also necessitate more complex differentiation, exacerbating optimization difficulties and computational burdens. The visualization results and the loss function are shown in FIGURE A1. It can be observed that the low accuracy of XPINNs is primarily caused by the failure to satisfy the boundary and interface conditions. In contrast, in AS-PINNs, the boundary and interface conditions are enforced through hard constraints, thus avoiding the cumbersome weight adjustments and optimization burden. Longer training or more weight adjustments might further improve XPINNs' accuracy, but this is sufficient to demonstrate the competitiveness of AS-PINNs.

Table A.1
Results of solving sixth-order differential equations using different methods.

NN	RAR	L_2 Relative Error						
		y	h	w	$w_1(\varphi)$	$w_2(M)$	$w_3(V)$	$w_4(Q)$
1 + 1+1 + 1+3 + 3 [a]	Yes	**2.57E-03**	**4.64E-03**	**4.25E-03**	**4.91E-03**	**7.56E-04**	2.02E-04	9.61E-05
1 + 1+1 + 1+3 + 3 [b]	Yes	2.26E-03	2.92E-03	3.04E-03	2.48E-03	4.25E-04	**2.42E-04**	**1.30E-04**
XPINNs	Yes	3.98E+01	3.27E+1	5.08E+00	8.08E-01	6.94E-01	4.18–01	2.13E-01
XPINNs	No	6.14E-01	4.68E-01	2.62E-01	1.20E-01	4.03E-02	9.27E-01	6.58E-01

[a] AS-PINNs with trainable transition points.
[b] AS-PINNs with fixed transition points.

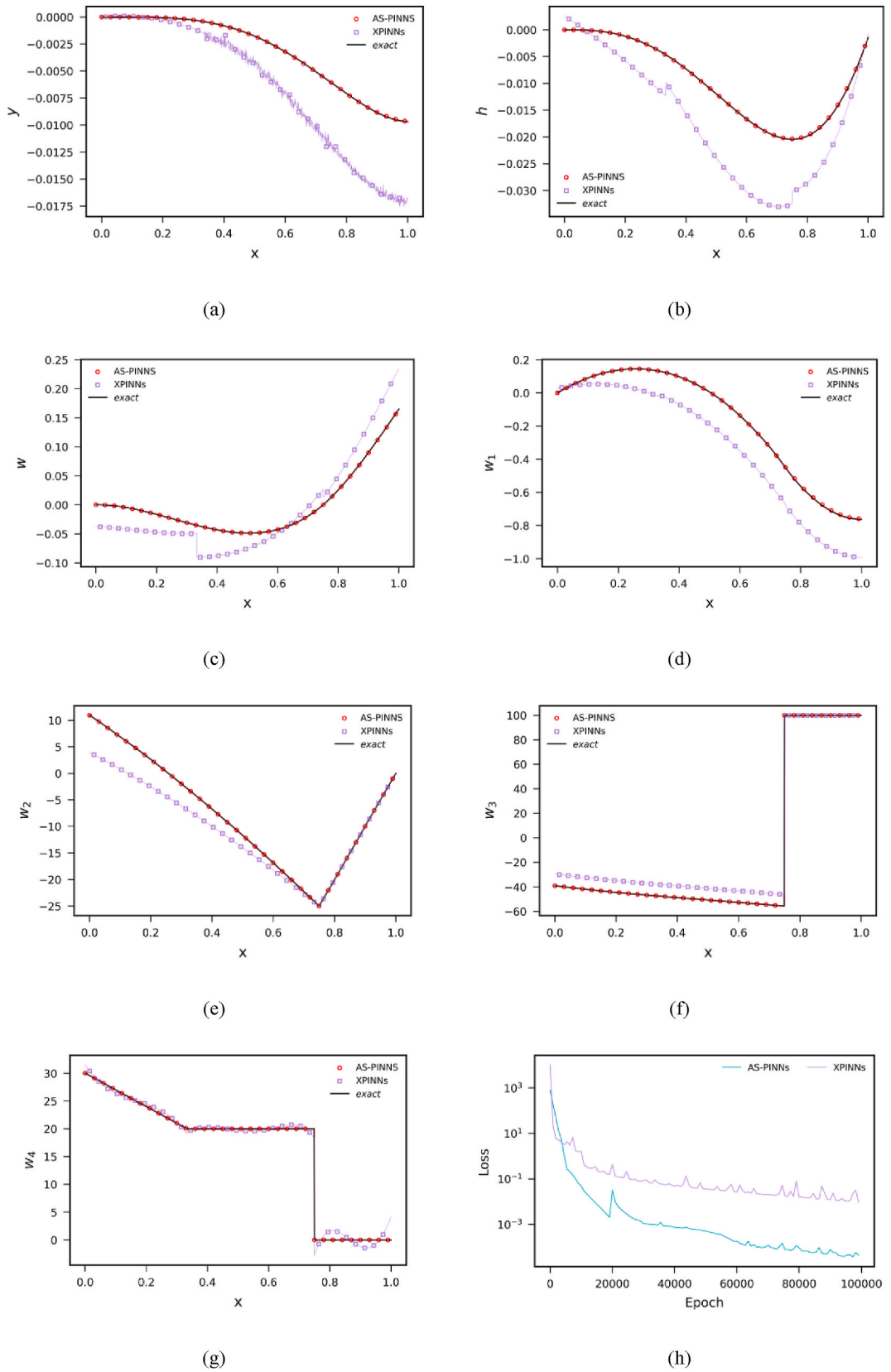


Fig. A.1. (a-g) Results of different physical variations of the sixth-order differential equations. (h) Loss function.

Data availability

I have shared the link to my data

References

Alipanahi, B., Delong, A., Weirauch, M.T., Frey, B.J., 2015. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. Nat. Biotechnol. 33, 831–838.

Antonio, F.-S., David, M.-G.J., Roberto, R.d.A., Alejandro, T.-F., Antonio, F.-R.J., 2024. Gradient-annihilated PINNs for solving riemann problems: application to relativistic hydrodynamics. Comput. Methods Appl. Mech. Eng. 424, 116906.

Bhowmick, S., Nagarajaiah, S., 2023. Physics-guided identification of Euler–Bernoulli beam PDE model from full-field displacement response with SimultaNeous basis function Approximation and Parameter Estimation (SNAPE). Eng. Struct. 289, 116231.

Bonkile, M.P., Awasthi, A., Lakshmi, C., Mukundan, V., Aswin, V., 2018. A systematic literature review of Burgers' equation with recent advances. Pramana 90, 1–21.

Cao, W., Zhang, W., 2024. An Analysis and Solution of Ill-Conditioning in Physics-Informed Neural Networks arXiv preprint arXiv:2405.01957.

Chang, Z., Li, K., Zou, X., Xiang, X., 2022. High order deep neural network for solving high frequency partial differential equations. Commun. Comput. Phys. 31, 370–397.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Math. Control, Signals, Syst. 2, 303–314.

Daubechies, I., 1992. Ten Lectures on Wavelets. Society for Industrial and Applied Mathematics, Philadelphia.

Diao, Y., Yang, J., Zhang, Y., Zhang, D., Du, Y., 2023. Solving multi-material problems in solid mechanics using physics-informed neural networks based on domain decomposition technology. Comput. Methods Appl. Mech. Eng. 413, 116120.

Faroughi, S., Darvishi, A., Rezaei, S., 2023. On the order of derivation in the training of physics-informed neural networks: case studies for non-uniform beam structures. Acta Mech. 234, 5673–5695.

Hao, Z., Su, C., Liu, S., Berner, J., Ying, C., Su, H., Anandkumar, A., Song, J., Zhu, J., 2024. Dpot: Auto-Regressive Denoising Operator Transformer for Large-Scale Pde Pre-training arXiv preprint arXiv:2403.03542.

Hou, J., Li, Y., Ying, S., 2023. Enhancing PINNs for solving PDEs via adaptive collocation point movement and adaptive loss weighting. Nonlinear Dynam. 111, 15233–15261.

Hu, W.-F., Lin, T.-S., Lai, M.-C., 2022. A discontinuity capturing shallow neural network for elliptic interface problems. J. Comput. Phys. 469, 111576.

Hu, Z., Jagtap, A.D., Karniadakis, G.E., Kawaguchi, K., 2023. Augmented Physics-Informed Neural Networks (APINNs): a gating network-based soft domain decomposition methodology. Eng. Appl. Artif. Intell. 126, 107183.

Jagtap, A.D., Karniadakis, G.E., 2020. Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Commun. Comput. Phys. 28, 2002–2041.

Jeong, H., Batuwatta-Gamage, C., Bai, J., Xie, Y.M., Rathnayaka, C., Zhou, Y., Gu, Y., 2023. A complete physics-informed neural network-based framework for structural topology optimization. Comput. Methods Appl. Mech. Eng. 417, 116401.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L., 2021. Physics-informed machine learning. Nature Reviews Physics 3, 422–440.

Kashefi, A., Guibas, L.J., Mukerji, T., 2023. Physics-informed PointNet: on How Many Irregular Geometries Can it Solve an Inverse Problem Simultaneously? Application to Linear Elasticity arXiv preprint arXiv:2303.13634.

Kendall, A., Gal, Y., Cipolla, R., 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7482–7491.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 60, 84–90.

Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B., 2015. Human-level concept learning through probabilistic program induction. Science 350, 1332–1338.

Li, X., Bolandi, H., Salem, T., Lajnef, N., Boddeti, V.N., 2022. NeuralSI: structural parameter identification in nonlinear dynamical systems. European Conference on Computer Vision. Springer Nature Switzerland, Cham, pp. 332–348.

Liu, L., Liu, S., Xie, H., Xiong, F., Yu, T., Xiao, M., Liu, L., Yong, H., 2023. Discontinuity computing using physics-informed neural networks. J. Sci. Comput. 98, 22.

Liu, Z., Cai, W., Xu, Z.-Q.J., 2020. Multi-ScaleDeepNeural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains. Commun. Comput. Phys. 28, 1970–2001.

Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E., 2021a. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat. Mach. Intell. 3, 218–229.

Lu, L., Meng, X., Mao, Z., Karniadakis, G.E., 2021b. DeepXDE: a deep learning library for solving differential equations. SIAM Rev. 63, 208–228.

Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., Johnson, S.G., 2021c. Physics-informed neural networks with hard constraints for inverse design. SIAM J. Sci. Comput. 43, B1105–B1132.

Luong, K.A., Le-Duc, T., Lee, J., 2023. Deep reduced-order least-square method—a parallel neural network structure for solving beam problems. Thin-Walled Struct. 191, 111044.

Luong, K.A., Le-Duc, T., Lee, S., Lee, J., 2024. A novel normalized reduced-order physics-informed neural network for solving inverse problems. Eng. Comput. 1–20.

Mattey, R., Ghosh, S., 2022. A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations. Comput. Methods Appl. Mech. Eng. 390, 114474.

McClenny, L.D., Braga-Neto, U.M., 2023. Self-adaptive physics-informed neural networks. J. Comput. Phys. 474, 111722.

Miao, Z., Chen, Y., 2023. VC-PINN: variable coefficient physics-informed neural network for forward and inverse problems of PDEs with variable coefficient. Phys. Nonlinear Phenom. 456, 133945.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707.

Saadat, M.H., Gjorgiev, B., Das, L., Sansavini, G., 2022. Neural Tangent Kernel Analysis of PINN for Advection-Diffusion Equation arXiv preprint arXiv:2211.11716.

Sarma, A.K., Roy, S., Annavarapu, C., Roy, P., Jagannathan, S., 2024. Interface PINNs (I-PINNs): a physics-informed neural networks framework for interface problems. Comput. Methods Appl. Mech. Eng. 429, 117135.

Subramanian, S., Kirby, R.M., Mahoney, M.W., Gholami, A., 2023. Adaptive self-supervision algorithms for physics-informed neural networks. ECAI 2023. IOS Press, Amsterdam.

Tseng, Y.-H., Lin, T.-S., Hu, W.-F., Lai, M.-C., 2023. A cusp-capturing PINN for elliptic interface problems. J. Comput. Phys. 491, 112359.

Vadeboncoeur, A., Akyildiz, Ö.D., Kazlauskaite, I., Girolami, M., Cirak, F., 2023. Fully probabilistic deep models for forward and inverse problems in parametric PDEs. J. Comput. Phys. 491.

Wang, H., Lu, L., Song, S., Huang, G., 2023. Learning Specialized Activation Functions for Physics-Informed Neural Networks arXiv preprint arXiv:2308.04073.

Wang, S., Yu, X., Perdikaris, P., 2022a. When and why PINNs fail to train: a neural tangent kernel perspective. J. Comput. Phys. 449, 110768.

Wang, Y., Sun, J., Li, W., Lu, Z., Liu, Y., 2022b. CENN: conservative energy method based on neural networks with subdomains for solving variational problems involving heterogeneous and complex geometries. Comput. Methods Appl. Mech. Eng. 400, 115491.

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

Yu, J., Lu, L., Meng, X., Karniadakis, G.E., 2022. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. Comput. Methods Appl. Mech. Eng. 393, 114823.

Yu, Y., Cai, C., Liu, Y., 2021. Probabilistic vehicle weight estimation using physics-constrained generative adversarial network. Comput. Aided Civ. Infrastruct. Eng. 36, 781–799.