A Hard Constraint and Domain-Decomposition-Based Physics-Informed Neural Network Framework for Nonhomogeneous Transient Thermal Analysis

Zengkai Wu, *Graduate Student Member, IEEE*, Li Jun Jiang, *Fellow, IEEE*, Sheng Sun, *Senior Member, IEEE*, and Ping Li, *Senior Member, IEEE*

***Abstract*— In this article, a hard constraint (HC) and domain-decomposition-based physics-informed neural network (HCD-PINN) framework is introduced for nonhomogeneous transient thermal analysis. In general, physics-informed neural network (PINN) uses a global neural network to approximate the solutions of partial differential equations (PDEs), and its performance could decrease dramatically when the problem becomes big or complex. To get this deficiency addressed and simultaneously enhance the modeling capability of PINN, in this work, the domain decomposition method (DDM)-based strategy is introduced. In each subdomain, an independent neural network is used to approximate the solution. Thereby, the size and complexity of the neutral network are reduced. To facilitate effective integration of solutions across different regions, an HC method is proposed for automatic satisfaction of interface conditions between adjacent subdomains. At the interface, continuity conditions for temperature and heat flux are considered, with heat flux continuity expressed in terms of the derivative of temperature. Using the mixed residual method (MIM), continuity conditions at the interface can be transformed into a linear form of the neural network outputs. This eliminates the need for differentiation, enabling automatic satisfaction of conditions through the use of a predefined HC matrix. Ultimately, we merge neural networks responsible for subdomains and interfaces, along with the HC matrix, using a differentiable distance function. This integration establishes a cohesive and unified framework. To validate the efficiency and accuracy of HCD-PINN, several numerical examples are studied and compared with previous PINN methods, with COMSOL simulations as exact solutions. The experimental results demonstrate the superior accuracy of our proposed method.**

***Index Terms*— 3-D packages, domain decomposition (DDM), hard constraint (HC), physics-informed neural network (PINN), transient thermal analysis.**

Manuscript received 15 May 2024; accepted 14 June 2024. Date of publication 19 June 2024; date of current version 16 August 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFA0709800, in part by NSFC under Grant 62071290, and in part by Shanghai Committee of Science and Technology under Grant 20501130500. Recommended for publication by Associate Editor L. Codecasa upon evaluation of reviewers' comments. *(Corresponding author: Ping Li.)*

Zengkai Wu and Ping Li are with the State Key Laboratory of Radio Frequency Heterogeneous Integration, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: ping.li@sjtu.edu.cn).

Li Jun Jiang is with the Department of Electrical Engineering, Missouri University of Science and Technology, Rolla, MO 65409 USA.

Sheng Sun is with the School of Electronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCPMT.2024.3416523.

Digital Object Identifier 10.1109/TCPMT.2024.3416523

I. Introduction

With the evolution process of semiconductor technology, the integration density of integrated circuits (ICs) is increasing continuously. Further fueled by the advanced 3-D packaging technologies [1], [2], more devices and modules can be integrated in a limited space, which leads to a significant increase in the heat density of ICs. The increasing heat density poses a serious challenge to thermal design, because uneven thermal managements would easily result in the temperature of the ICs being too high, while not only affecting the performance and service life of the system but also compromising the reliability of the system. Therefore, accurate and robust thermal analysis of ICs is crucial for their reliable operation. To facilitate the IC design, accurate transient thermal analysis is essential.

To accurately and efficiently facilitate the thermal analysis, various numerical methods have been proposed in the past years. Finite element time-domain (FETD) method [3], [4], [5] is a widely used method for thermal analysis, which discretizes the domain into small elements and solves the heat transfer equation for each element using the time-domain approach, such as Euler method [4], generalized-α method [6], leap frog [7], and Runge–Kutta [8]. The spectral-element time-domain (SETD) method [9], [10], [11] can be viewed as a special case of FETD method, which discretizes solution domain with hexahedron and mainly uses high-order polynomials to implement the interpolation [11]. The discontinuous Galerkin (DG) method [12], [13], [14], [15] combines the finite-element method (FEM) and finite-volume method (FVM). A term called "numerical flux" [16] is introduced in DG for the communication of two adjacent elements, which allows the DGTD method to model arbitrary shapes and to achieve high-order accuracy.

In recent years, unlike traditional numerical methods [3], [4], [5], [9], [10], [12], [13], [14], a novel deep learning method termed as physics informed neural network (PINN) [17], [18] has been proposed to solve nonlinear PDEs without spatial and temporal discretization. It uses automatic differentiation [19] as a differential operator to represent the derivatives, avoiding discretization.

In PINN, a fully connected neural network (FCNN) is used as the surrogate model for the solution of PDEs. The model can be trained to approximate the solutions exactly using the optimizer [20], [21] to minimize the sum of loss terms of conditions in PDEs, which are governing equations, boundary conditions, and initial conditions. PINN has been proven to be effective in many fields such as flow mechanics [22], [23], [24], material design [25], [26], heat transfer analysis [27], [28], and geospatial modeling [29], [30].

For the traditional PINN, an FCNN is used as the surrogate model to approximate the solution of a PDE. Usually, the outputs and the gradients of the FCNN are continuous. However, for nonhomogeneous coefficient problems, due to the discontinuous solutions and gradients, the approximation error becomes obvious. Thus, to enhance the capability of PINN in addressing complex structures and irregular domains, various approaches based on the domain decomposition method (DDM) have been proposed. Through tearing the entire domain into a number of small subdomains, the complexity of the PINN neural network is alleviated significantly. For instance, the cPINN in [31] and the XPINN in [32] split the whole domain into a set of nonoverlapping subdomains, while the FBPINN [33] divides the solution domain into multiple overlapping subdomains. To connect the solutions among neighboring subdomains, the continuity conditions across the interface are deployed as the constraints. These methods unite the solutions of entire domain by introducing new loss functions at the interfaces, which requires the neural network to learn extra loss conditions and thus increase the training cost [34].

In the traditional PINN formulations, the boundary conditions (BCs) and the initial conditions (ICs) are mainly incorporated into the training process of neural networks through the form of loss functions, known as the soft constraints. Extensive researches [34], [35], [36], [37] make efforts to integrate the boundary conditions into the neural network solutions by modifying the structure of the neural networks, which is referred as the hard constraints (HCs). In [35], an HC method with a Ritz variation formulation is proposed and integrated with the free neural network (PFNN) method. In [37], an HC method is used to solve the forward and inverse problems. Notably, a unified framework called HC-Net [34] has been proposed to implement three common types of boundary conditions using HCs, such as Dirichlet BC, Neumann BC, and Robin BC. The HC [34], [36] not only reduces the error in boundary conditions but also allows the neural network to focus on learning the remaining equations, thereby improving the performance of the neural network.

In general, thermal analysis is characterized by complex structures and diverse material compositions. When using traditional PINN for computations, significant errors are often present. In this article, we proposed an HC for the interface condition enhanced domain decomposition PINN framework termed as hard constraint and domain-decomposition-based physics-informed neural network (HCD-PINN) to solve the discontinuous and multiple coefficient thermal analysis problems. Similar to the previously mentioned DDMs [31], [32], the solution domain is divided into several nonoverlapping subdomains, and the solution in each individual subdomain is expressed using an independent neural network. This scheme reduces the scale of each neural network, mitigating the training difficulty. To ensure the continuous property of solutions across the interface, in this work, the HC [34] strategy is resorted to. In contrast to the traditional DDMs that introduce loss terms to achieve this continuity, the HC method reduces the number of loss terms in the neural network learning, thus enhancing the accuracy and efficiency. The main idea of HC is to introduce a new neural network for each interface and automatically satisfy continuity through a derived matrix. The primary challenge lies in the evaluation of the spatial derivatives of temperature for the heat flux continuity at the interfaces. Therefore, we use the mixed residual method (MIM) [38] method to reconstruct the outputs of neural network, enabling the linear implementation of the equation for heat flux continuity. The forthcoming sections of this work will explicate the derivation process of HC matrix. Finally, a continuously differentiable distance function is introduced to combine the neural networks for subdomains and interface networks, achieving an integrated structure.

Through comparative experiments, we observed that the proposed framework is capable of addressing complex problems. Compared with the traditional PINN and DDM-based PINN (XPINN), the proposed HCD-PINN can achieve higher accuracy. As for the geometrically complex problem, PINN requires a huge neural network to represent the solutions of the PDEs, which is hard to train. However, the proposed method needs smaller neural networks and fewer parameters to approximate solutions, thus free of training difficulties.

The rest of this article is organized as follows. Section II briefly introduces the formulation of problem including transient thermal analysis and structure of PINN. Section III presents the formulation of HCD-PINN. Several numerical results are shown in Section IV to demonstrate the effectiveness of the proposed method. Specific conclusions are summarized in Section V.

II. Transient Thermal Analysis with Traditional PINNs

A. Transient Heat Transfer Problem

Assume that the domain of interest (DOI) for the transient thermal analysis is denoted as $\mathbf{\Omega}$ with $\partial\mathbf{\Omega}$ representing the boundary of DOI.

The governing PDEs of the transient heat equation are defined as

$$\rho C_p \frac{\partial T(\boldsymbol{x},t)}{\partial t} = \nabla\cdot\kappa\nabla T(\boldsymbol{x},t) + Q, \quad \boldsymbol{x}\in\mathbf{\Omega} \tag{1}$$

$$\boldsymbol{n}(\boldsymbol{x})\cdot\kappa\nabla T(\boldsymbol{x},t) = -h(T(\boldsymbol{x},t)-T_a), \quad \boldsymbol{x}\in\partial\mathbf{\Omega} \tag{2}$$

$$T(\boldsymbol{x},0) = T_0, \quad \boldsymbol{x}\in\mathbf{\Omega}. \tag{3}$$

In the above formula, (1) is the heat equation, (2) and (3) are the convective boundary condition and the initial condition, respectively; Q, ρ, C_p, and κ denote the heat source, the density, the heat capacity, and the thermal conductivity of the material, respectively; T_a and T_0 indicate the ambient and initial temperature of the problem, respectively, and h represents the convection coefficient of the boundary condition.

B. PINN Method for Transient Thermal Problem

In this work, PINNs use an FCNN to approximate the solution of transient thermal equation. Namely,

$$\hat{T}(\boldsymbol{x}, t) = \mathrm{NN}(\boldsymbol{x}, t; \boldsymbol{\theta}) \tag{4}$$

where $\hat{T}(\boldsymbol{x}, t)$ is the approximated solution, $\mathrm{NN}(\boldsymbol{x}, t; \boldsymbol{\theta})$ is the output of FCNN, $(\boldsymbol{x}, t)$ is the input of the network in the spatial and time domains, and $\boldsymbol{\theta}$ is the parameter of the network.

The neural network includes an input layer, an output layer, and a certain number of hidden layers. The forward pass between two layers is expressed as

$$\boldsymbol{y}_i = \sigma(\boldsymbol{w}_i \cdot \boldsymbol{x}_i + \boldsymbol{b}_i) \tag{5}$$

where $\boldsymbol{y}_i$ and $\boldsymbol{x}_i$ denote the input and output of the ith layer, respectively; $\boldsymbol{w}_i$ and $\boldsymbol{b}_i$ are the weights and biases of the ith layer, which are trainable parameters; $\sigma(\cdot)$ is a nonlinear activation function, and the choice of $\sigma(\cdot)$ will be discussed later in Section IV-C.

The parameters of the neural network can be trained by minimizing the loss function defined as

$$\mathcal{L} = w_g \cdot \mathcal{L}_g + w_b \cdot \mathcal{L}_b + w_i \cdot \mathcal{L}_i \tag{6}$$

with

$$\mathcal{L}_g = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\rho C_p \frac{\partial \hat{T}(\boldsymbol{x}_i, t_i)}{\partial t} - \nabla \cdot \kappa \nabla \hat{T}(\boldsymbol{x}_i, t_i) - Q\right|^2 \tag{7}$$

$$\mathcal{L}_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|\kappa \frac{\partial \hat{T}(\boldsymbol{x}_i, t_i)}{\partial \boldsymbol{n}} + h(\hat{T}(\boldsymbol{x}_i, t_i) - T_a)\right|^2 \tag{8}$$

$$\mathcal{L}_i = \frac{1}{N_i}\sum_{i=1}^{N_i}\left|\hat{T}(\boldsymbol{x}_i, 0) - T_0\right|^2 \tag{9}$$

where N_f, N_b, and N_i represent the number of sampling points pertinent to the governing equations, the boundary conditions, and the initial data, respectively; and $\mathcal{L}_g$, $\mathcal{L}_b$, and $\mathcal{L}_i$ are the loss terms corresponding to the heat governing equation, the boundary conditions, and the initial conditions, respectively. The loss weights w_g, w_b, and w_i determine the relative importance of different terms. The role of loss weights is primarily to balance the values of different loss terms to ensure balanced convergence [41]. In this article, we estimated the magnitude of each loss term in advance and determined loss weights to make the values of loss terms relatively close. Across all the studies in this article, the same weights are used for different methods to minimize the influence of weights on the results and purely compare the impact of neural network structures on accuracy. The impact of loss weight on computational accuracy will be discussed in Section IV-C.

When solving time-domain problems, unlike the traditional computational methods, PINN only requires adding time as an additional dimension to the neural network's training datasets. Moreover, PINN is not sensitive to dimensions, making it well-suited for solving high-dimensional problems. However, for transient analysis, PINN does encounter certain issues, such as the error accumulation over time and difficulties in predicting the "sharpness" of the solution [39]. Efforts have been proposed to overcome these problems, such as "sequence to sequence" [39] learning and decomposition in the time domain [40].

Traditionally, the main issue arises when the time span is too long, whereas most activation functions are more adept at computing inputs within the range of $[-1, 1]$. To address this deficiency, we normalize the time domain as

$$t \in [0, t_0] \rightarrow t' \in [0, 1] \tag{10}$$

and thus, the derivative of temperature with respect to time can be transformed as

$$\frac{\partial \hat{T}(\boldsymbol{x}, t')}{\partial t'} = t_0 \frac{\partial \hat{T}(\boldsymbol{x}, t)}{\partial t}. \tag{11}$$

By normalizing the time domain, the PINN can solve the transient problems much more accurately.

III. Formulation of Proposed HCD-PINN

A. HC for Initial Condition

For transient heat problems with known initial conditions, when using PINNs, the HC method [34] incorporates the initial conditions into the estimated solution as

$$\hat{T}(\boldsymbol{x}, t) = \tilde{T}(\boldsymbol{x}, t)(1 - \exp[-\gamma t]) + T_0(\boldsymbol{x}) \tag{12}$$

where γ is a hyperparameter for initial condition. $\tilde{T}(\boldsymbol{x}, t)$ is the output of the neural network. $T_0(\boldsymbol{x})$ is usually given as the initial condition for transient thermal analysis. $(1-\exp[-\gamma t])$ ensures that at $t = 0$ the estimated solution is always equal to $T_0(\boldsymbol{x})$, and the larger γ causes the initial conditions satisfy better. However, too large γ may cause the derivation of $(1 - \exp[-\gamma t])$ with respect to t too sharp, which will deteriorate the accuracy of the neural network. Therefore, in this article, γ is set to 10.

Using (12), the initial conditions are automatically satisfied in the solution process. Consequently, the loss function (9) can be eliminated. Thereby, the adoption of the HC method not only reduces the training burden on the neural network, but also makes the training process focus more on the other loss components.

B. Domain Decomposition Method

To solve the heat transfer problem with different materials, a DDM is introduced first. The main idea of DDM is to divide the solution domain into several nonoverlapping subdomains according to the type of material, ensuring that a single neural network is built for each media

$$\boldsymbol{\Omega} = \boldsymbol{\Omega}^1 \cup \boldsymbol{\Omega}^2 \cup \cdots \cup \boldsymbol{\Omega}^M. \tag{13}$$

An independent neural network is used to approximate the solution of each subdomain as

$$\mathrm{NS}^k(\boldsymbol{x},t)=\mathrm{NN}^k_{\mathrm{sub}}\left(\boldsymbol{x},t;\boldsymbol{\theta}^k_{\mathrm{sub}}\right):\mathbb{R}^{d+1}\to\mathbb{R}^{d+1} \tag{14}$$

where d is the dimension of heat transfer problems. $\mathrm{NS}^k(\boldsymbol{x},t)$ means the output of NN for subdomain $\boldsymbol{\Omega}^k$. The input dimension of NN is $d+1$, including the spatial coordinates $\boldsymbol{x}$ and time t. The output dimension is $d+1$, including the solution of heat transfer problems $T(\boldsymbol{x},t)$ and its temperature gradients $\boldsymbol{p}(\boldsymbol{x},t)$, which will be discussed in Section III-C.

C. HC for Interface Conditions

Suppose that $\partial\boldsymbol{\Omega}^{kl}$ denotes the interface between two neighboring subdomains $\boldsymbol{\Omega}^k$ and $\boldsymbol{\Omega}^l$. At the subdomain interface $\partial\boldsymbol{\Omega}^{kl}$, the continuity conditions are defined as

$$T^k-T^l=0 \tag{15}$$

$$\boldsymbol{n}^{kl}(\boldsymbol{x})\cdot\boldsymbol{q}^k+\boldsymbol{n}^{lk}(\boldsymbol{x})\cdot\boldsymbol{q}^l=0 \tag{16}$$

where $\boldsymbol{n}^{kl}(\boldsymbol{x})$ and $\boldsymbol{n}^{lk}(\boldsymbol{x})$ represent the unit normal vector on the interface $\partial\boldsymbol{\Omega}^{kl}$ from $\boldsymbol{\Omega}^k$ to $\boldsymbol{\Omega}^l$ and from $\boldsymbol{\Omega}^l$ to $\boldsymbol{\Omega}^k$, respectively; The variable $\boldsymbol{q}$ denotes the heat flux given by

$$\boldsymbol{q}^k=-\kappa^k\nabla T^k,\quad \boldsymbol{x}\in\boldsymbol{\Omega}^k. \tag{17}$$

The temperature continuity at the interface between subdomains $\boldsymbol{\Omega}^k$ and $\boldsymbol{\Omega}^l$ is enforced by (15), while the continuity of the normal component of the heat flux density is enforced by (16).

According to [38], an extra field $\boldsymbol{p}(\boldsymbol{x},t)$ is introduced as a part of the solution to (1)

$$\boldsymbol{p}(\boldsymbol{x},t)=(p_1(\boldsymbol{x},t),\ldots,p_d(\boldsymbol{x},t))=\nabla T(\boldsymbol{x},t) \tag{18}$$

where $T_N(\boldsymbol{x},t)=(T(\boldsymbol{x},t),\boldsymbol{p}(\boldsymbol{x},t))$ is the new solution of (1).

By applying (18) to (16), the continuity conditions can be converted into the linear combinations of $T_N(\boldsymbol{x},t)$, which are given by

$$\kappa^k(\hat{\boldsymbol{n}}^{kl}\cdot\boldsymbol{p}^k(\boldsymbol{x},t))+\kappa^l(\hat{\boldsymbol{n}}^{lk}\cdot\boldsymbol{p}^l(\boldsymbol{x},t))=0. \tag{19}$$

Instead of introducing new loss functions to learn (15) and (19), we use HC to make them satisfied automatically [34]. Because the interface conditions have been transformed into linear combinations of $T_N(\boldsymbol{x},t)$, we can use matrix manipulation to combine the outputs of NN.

First, for each interface, an additional NN is proposed for HC as

$$\mathrm{NI}^{kl}(\boldsymbol{x},t)=\mathrm{NN}^{kl}_{\mathrm{inter}}(\boldsymbol{x},t;\theta^{kl}_{\mathrm{inter}}):\mathbb{R}^{d+1}\to\mathbb{R}^m \tag{20}$$

where $\mathrm{NI}^k(\boldsymbol{x},t)$ means the output of NN for interface $\partial\boldsymbol{\Omega}^{kl}$; $d+1$ represents the input dimension, including the spatial coordinates $\boldsymbol{x}$ and the time t; and m denotes the output dimension, which is equal to the number of nonzero elements d_0 in the normal vector of the interface. For example, if the interface is perpendicular to the x-axis, then its normal vector is $\boldsymbol{n}=(1,0,0)$, and the number of nonzero elements is denoted as $d_0=1$. The relationship between m and d_0 is

$$m=\begin{cases}2d_0, & d_0=1,2\\ 2d_0+2, & d_0\geq 3.\end{cases} \tag{21}$$

Then, we introduce a pair of HC matrices $\boldsymbol{B}^i(\boldsymbol{x})\in\mathbb{R}^{d+1}\times\mathbb{R}^m$ to reconstruct $\mathrm{NI}^{kl}(\boldsymbol{x},t)$. Namely

$$\hat{T}^{kl}_N(\boldsymbol{x},t)=\boldsymbol{B}^+(\boldsymbol{x})\mathrm{NI}^{kl}(\boldsymbol{x},t) \tag{22}$$

$$\hat{T}^{lk}_N(\boldsymbol{x},t)=\boldsymbol{B}^-(\boldsymbol{x})\mathrm{NI}^{kl}(\boldsymbol{x},t). \tag{23}$$

The interface conditions in (15) and (19) can be satisfied automatically by the combination of the additional neural network in (20) and a well-selected HC matrix $\boldsymbol{B}^i(\boldsymbol{x})$.

If $d_0=1$, the HC matrices can be obtained easily as

$$\boldsymbol{B}^+(\boldsymbol{x})=\begin{bmatrix}1&0\\0&1\end{bmatrix} \tag{24}$$

$$\boldsymbol{B}^-(\boldsymbol{x})=\begin{bmatrix}1&0\\0&\kappa^+(\boldsymbol{x})/\kappa^-(\boldsymbol{x})\end{bmatrix}. \tag{25}$$

Similarly, if $d=2$, the HC matrix should be implemented with $m=4$ linearly independent vectors. The HC matrix is expressed as

$$\boldsymbol{B}^+(\boldsymbol{x})=\begin{bmatrix}1&0&0&0\\0&\hat{n}_2&0&\hat{n}_1\\0&-\hat{n}_1&0&\hat{n}_2\end{bmatrix} \tag{26}$$

$$\boldsymbol{B}^-(\boldsymbol{x})=\begin{bmatrix}1&0&0&0\\0&0&\hat{n}_2&\kappa_1\hat{n}_1\\0&0&-\hat{n}_1&\kappa_1\hat{n}_2\end{bmatrix} \tag{27}$$

where $\kappa_1=\kappa^+(\boldsymbol{x})/\kappa^-(\boldsymbol{x})$ is the contrast of the coefficients $\kappa(\boldsymbol{x})$.

The general solution for $d_0\geq 3$ problem is detailed in Appendix A.

D. Smoothly Gradient Distance Function

In this section, the smoothly gradient distance functions $\mathcal{D}(\cdot)$ are introduced to connect the proposed $\mathrm{NS}^k(\boldsymbol{x},t)$ and $\mathrm{NI}^{kl}(\boldsymbol{x},t)$. They are expressed as

$$\mathcal{D}_1(l)=\begin{cases}0, & l\leq-\beta\\ \dfrac{(l+\beta)^2}{4\beta}, & -\beta<l\leq\beta\\ l, & \beta<l\leq 1-\beta\\ \dfrac{(1+\beta-l)^2}{4\beta}, & 1-\beta<l\leq 1+\beta\\ 1, & 1+\beta<l\end{cases} \tag{28}$$

and

$$\mathcal{D}_2(l)=\begin{cases}1, & l\leq-\beta\\ 1-\dfrac{(l+\beta)^2}{4\beta}, & -\beta<l\leq\beta\\ 1-l, & \beta<l\leq 1-\beta\\ 1-\dfrac{(1+\beta-l)^2}{4\beta}, & 1-\beta<l\leq 1+\beta\\ 0, & 1+\beta<l\end{cases} \tag{29}$$

where l is the normalized distance between the points $\boldsymbol{x}$ and interfaces; β is a hyperparameter, which controls the shape of these two types of distance functions. In general, the value of β is chosen as small as possible. However, a too small β makes the gradients of distance functions too sharp, which will deteriorate the accuracy. In this article, β is set as 0.01.

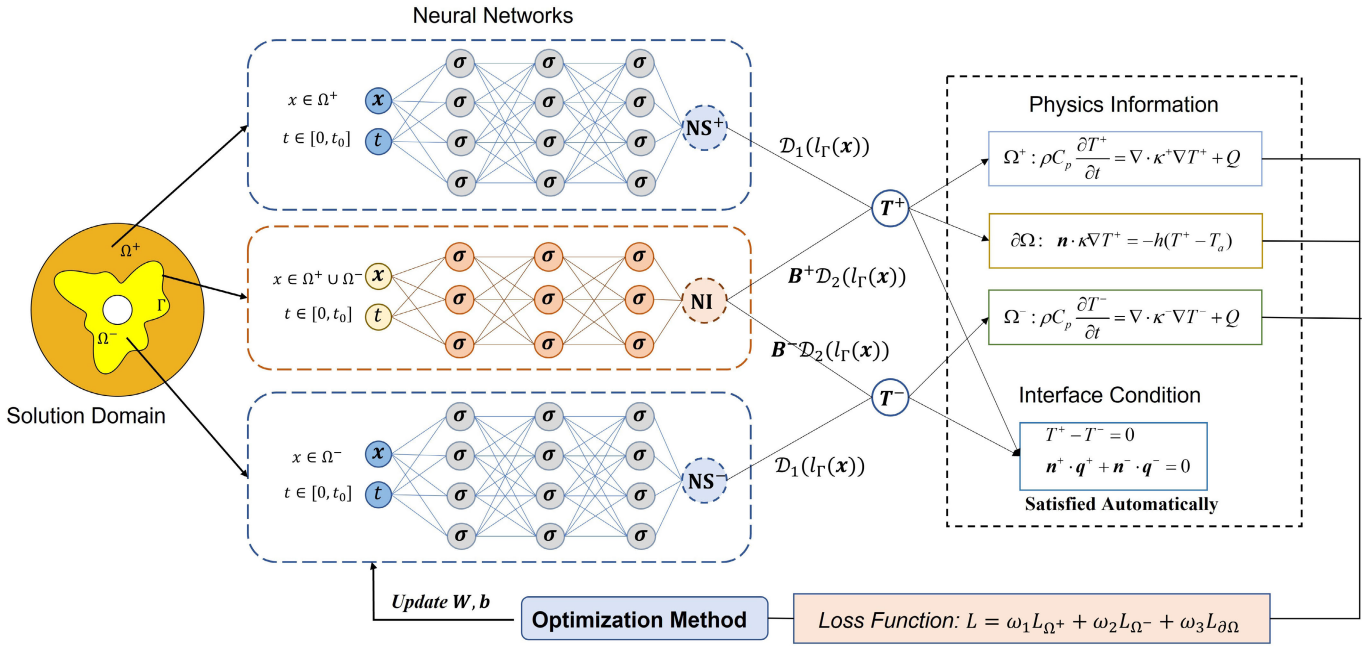

Fig. 1. Schematic of the proposed DHC-PINN framework. The structure of the schematic is inspired by [43]. The solution domain is divided into several subdomains based on materials. Neural networks for subdomain NS^+ and NS^- are introduced to approximate the solution in Ω^+ and Ω^-, respectively. Neural network NI is introduced for interface conditions. Note that although NI is served for interface conditions, the input spatial coordinates $\boldsymbol{x}$ should be the two subdomains adjacent to the interface $\boldsymbol{x} \in \Omega^+ \cup \Omega^-$. Then, the aforementioned two types of neural networks are combined for each subdomain's solution through distance functions $\mathcal{D}(\cdot)$ and HC matrix $\boldsymbol{B}(\boldsymbol{x})$.

The distance functions $\mathcal{D}_1(l)$ and $\mathcal{D}_2(l)$ are served as the weights of $\mathrm{NS}(\boldsymbol{x}, t)$ and $\mathrm{NI}(\boldsymbol{x}, t)$, respectively. Since $\mathrm{NS}(\boldsymbol{x}, t)$ is merely responsible for the approximation inside the corresponding subdomain, to ensure strict satisfaction of interface conditions, its weight at the interface is set to be 0. On the counterpart, $\mathrm{NI}(\boldsymbol{x}, t)$ primarily contributes to the solution at the interfaces, with its weight gradually decreasing as the distance from the interfaces increases.

The proposed distance functions proves to be more performing than the one presented in [34]. The detailed experiments are shown in Appendix B.

E. Framework of the Proposed HCD-PINN

Based on the theories listed above, the solution of heat transfer problems in subdomain $\boldsymbol{\Omega}^k$ can be constructed via the PINN as

$$\hat{T}_N^k(\boldsymbol{x}, t) = \mathcal{D}_1\left(l_{\text{inter}}^k(\boldsymbol{x})\right)\mathrm{NS}^k\left(\boldsymbol{x}, t; \theta_{\text{sub}}^k\right) + \sum_{i=1}^{m_k} \mathcal{D}_2\left(l_{\text{inter}}^i(\boldsymbol{x})\right)\boldsymbol{B}^i(\boldsymbol{x})\mathrm{NI}^i\left(\boldsymbol{x}, t; \theta_{\text{inter}}^i\right) \tag{30}$$

where $l_{\text{inter}}^i(\boldsymbol{x})$ with $i = 1, 2, \ldots, m_k$ denotes the distance to interfaces of $\boldsymbol{\Omega}^k$ and its m_k neighbors.

For better understanding, the PINN-based neural network framework structure for the thermal analysis is shown in Fig. 1.

Due to the application of the HC for the initial conditions, the loss function (9) pertinent to the initial conditions can be eliminated automatically. Thus, in subdomain k, the left parameters of the proposed framework can be trained by minimizing the loss function defined as below

$$\mathcal{L} = \sum_{k=1}^{M}\left(w_g \cdot \mathcal{L}_g^k + w_b \cdot \mathcal{L}_b^k + w_e \cdot \mathcal{L}_e^k\right). \tag{31}$$

with

$$\mathcal{L}_g^k = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\rho^k C_p^k \frac{\partial \hat{T}^k(\boldsymbol{x}_i, t)}{\partial t} - \nabla \cdot \kappa^k \hat{\boldsymbol{p}}^k(\boldsymbol{x}_i, t) - Q^k\right|^2 \tag{32}$$

$$\mathcal{L}_b^k = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|\kappa^k(\hat{n}(\boldsymbol{x}_i) \cdot \hat{\boldsymbol{p}}^k(\boldsymbol{x}_i, t)) + h^k(\hat{T}^k(\boldsymbol{x}_i, t) - T_a)\right|^2 \tag{33}$$

$$\mathcal{L}_e^k = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\left|\hat{\boldsymbol{p}}^k(\boldsymbol{x}_i, t) - \nabla \hat{T}^k(\boldsymbol{x}_i, t)\right|\right|_2^2. \tag{34}$$

where N_f and N_b represent the number of residual points sampled in $\boldsymbol{\Omega}^k$ and the number of boundary points, respectively; $\mathcal{L}_g^k$, $\mathcal{L}_b^k$, and $\mathcal{L}_e^k$ penalize the residuals of heat governing equations, the boundary conditions, and the extra field (18) in subdomain Ω_k, respectively.

IV. Numerical Results

In this section, two numerical results comparing HCD-PINN and PINN method [17] and DDM-based PINN method (XPINN) [32] are shown to demonstrate the effectiveness and accuracy of the proposed method.

The implementation of each method mentioned in this section and the training of neural networks were conducted using PyTorch [42]. The geometries of all the examples in this section are modeled with Deepxde toolbox [18]. The residual points, initial points, and boundary points used in training are selected randomly through the pseudorandom algorithm in Deepxde toolbox. To better explore the solution space [44], a strategy for reselecting training points every 20 epochs is adopted during the training process for each method.

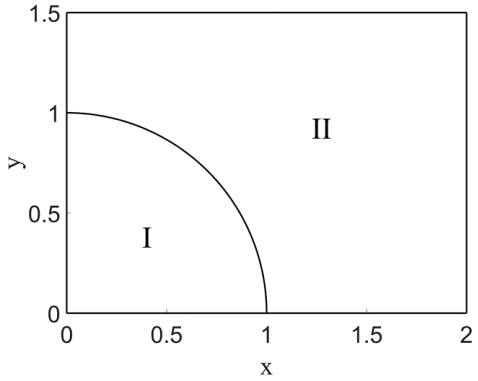

Fig. 2. Profile of the regions for Example I.

TABLE I
MATERIAL PROPERTY DETAILS OF THE FIRST EXAMPLE

Region	$\kappa\,(\mathrm{W}/[\mathrm{m}\cdot\mathrm{K}])$	$C_p\,(\mathrm{J}/[\mathrm{kg}\cdot\mathrm{K}])$	$\rho\,(\mathrm{kg}/[\mathrm{m}^3])$
I	2	2	2
II	1	1	1

For accuracy verification, the reference results are calculated by a finite-element-method-based commercial software COMSOL Multiphysics.

In this article, all the neural networks are trained with two optimization methods, Adam optimizer [20] and second-order L-BFGS optimizer [21]. When the loss function is optimized to be small enough with Adam optimizer, we use the quasi-Newton, full-batch gradient-based L-BFGS optimizer for further optimization. All the experiments are conducted on a desktop with Intel 3.6-GHz Core (TM) i9 CPU and 16-GB RAM under Windows Operating System. For convenience, we define the structure of NN (4) used in this article as: $[a] + [b] \times c + [d]$, which means that the NN has input layers with a neurons, c hidden layers with b neurons in each hidden layer, and output layer with d neurons.

A. Two-Dimensional Heat Convection with Rectangle and Semicircular Domain

For the first example, a theoretical 2-D model composed of two different materials is investigated, which is specifically designed to validate the effectiveness of the proposed method in addressing 2-D inhomogeneous problems. As illustrated in Fig. 2, the whole domain is a rectangular region divided by a quarter circle into two parts. The dimensions of the rectangular region are 2×1.5 m, and the radius of the circle is $r = 1$ m. The materials in Regions I and II are totally different, where the material parameters are listed in Table I.

Within the entire domain, a constant heat source defined as $Q = 50\ \mathrm{W/m^3}$ is introduced. The boundaries of the region are subjected to the convective heat condition with $h = 1\ \mathrm{W/m^2}$. We conducted computations to analyze the temperature variation process in $t \in [0, 10]$ s under the initial temperature $T_0 = 300$ K.

We compared the performance of three methods in solving this problem: PINN [17], XPINN [32], and the proposed method. The details of comparative experiments are shown as follows.

TABLE II
COMPARATIVE EXPERIMENTAL RESULT FOR EXAMPLE I

	Mean Absolute Percentage Error (MAPE) of T				Param
	$\mathrm{t} = 0\,\mathrm{s}$	$\mathrm{t} = 5\,\mathrm{s}$	$\mathrm{t} = 10\,\mathrm{s}$	Average	
PINN	0.393%	1.727%	1.671%	1.912%	15.0K
XPINN	0.110%	25.297%	26.711%	21.741%	13.5K
HCD-PINN	**0.002%**	**0.266%**	**0.219%**	**0.260%**	**11.8K**

1) *PINN: Original PINN Method* The whole solution domain is approximated by a single neural network. In this method, the number of residual points is $N_f = 8192$, the number of boundary points is $N_b = 2048$, and the number of initial points is $N_i = 2048$. The structure of NN is $[3] + [60] \times 5 + [1]$.
2) *XPINN: Original XPINN Method* The solution domain is divided into two subdomains, and two independent NN are used for each subdomain. In this method, $N_f = 4096$, $N_b = 1024$, $N_i = 1024$, and $N_{\text{inter}} = 512$ extra points are collocated for XPINN. The structure of NN in each subdomain is $[3] + [40] \times 5 + [1]$.
3) *HCD-PINN: Proposed HCD-PINN Method* The solution domain is divided into two subdomains. In this method, $N_f = 4096$, and $N_b = 1024$. The structure of NN in this method is NI:$[3] + [30] \times 5 + [3]$, and NS:$[3] + [30] \times 5 + [4]$.

In the COMSOL software, $N_t = 20\,385$ points and 40 271 triangle meshes with second-order Lagrange elements are tested as the standard solutions. The computation time for COMSOL is about 136 s. In this example, the loss weights are set as $(w_G, w_{BC}, w_E) = (1, 1, 1)$.

To avoid the "division by zero issue" and simultaneously enhance the contrast of the experimental results, we choose $T_1 = 273.15$ K as the reference temperature and calculate the mean absolute percentage error (MAPE) as

$$\mathrm{MAPE} = \frac{\sum_{i=1}^{n} \left|\hat{T}_i - T_i\right|}{\sum_{i=1}^{n} (T_i - 273.15\ \mathrm{K})} \tag{35}$$

where n is the number of testing points generated from COMSOL Simulations; $\hat{T}_i$ and T_i are obtained from NNs and COMSOL, respectively. The experimental result of Example I is shown in Table II. In this table, the MAPE of T and the number of trainable parameters in each method are compared. It is observed that the accuracy of the proposed method is significantly higher than those from other methods with fewer trainable parameters.

Table III gives the training and the prediction times for each method. The first and second columns show the training times for one Adam and L-BFGS epoch, respectively; the third column shows the total training time and the fourth column gives the prediction time for one point. It can be observed that the HCD-PINN method does not significantly increase training time compared with the other methods. However, in terms of the prediction time, there is a certain increase due to the

TABLE III
TRAINING AND PREDICT TIME FOR EXAMPLE I

	Adam (s)	L-BFGS (s)	Training (s)	Prediction (ms)
PINN	0.086	1.785	2043	0.21
XPINN	0.075	1.503	1728	0.18
HCD-PINN	0.091	1.724	1997	0.97

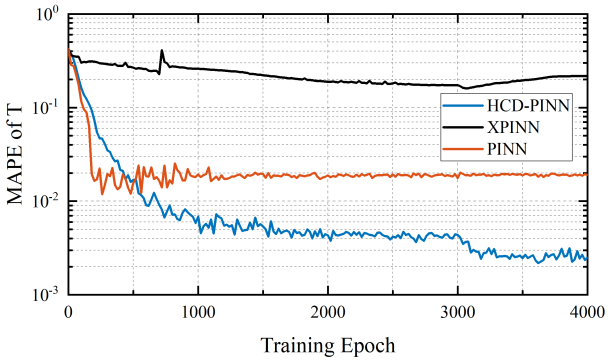

Fig. 3. Trajectories of relative error of T in Example I as a function of training epoch for PINN, XPINN, and HCD-PINN.

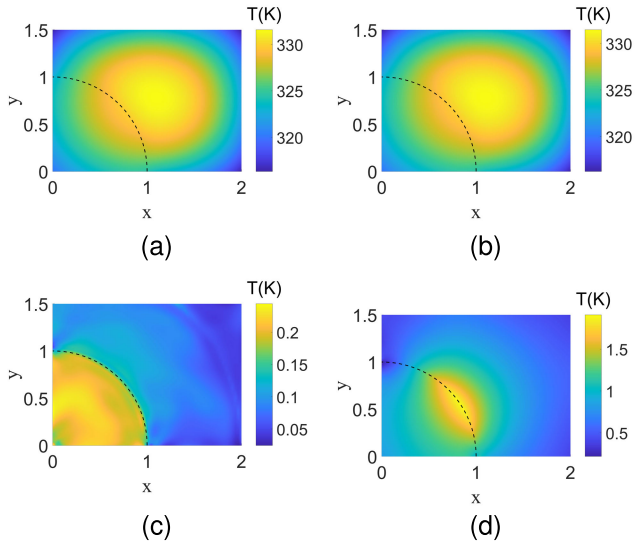

Fig. 4. Temperature distribution (unit: K) at $t = 10$ s by (a) COMSOL and (b) HCD-PINN. Error distribution (unit: K) with respect to COMSOL results at $t = 10$ s by (c) HCD-PINN and (d) PINN.

necessity of combining outputs from multiple neural networks into the HCD-PINN.

To validate that all the methods are able to converge during the training, the variations in MAPE with respect to T are presented in Fig. 3. Notably, the proposed method achieves the minimum relative error, which is further emphasized in Fig. 4. In Fig. 4, the temperature distributions solved by COMSOL and the proposed method at $t = 10$ s are plotted. In addition, the absolute differences between our proposed method, the PINN method, and the COMSOL are also given in Fig. 4. It can be seen that the maximum temperature error for DHC-PINN is 0.244 K, while the maximum error for the PINN method is 1.909 K.

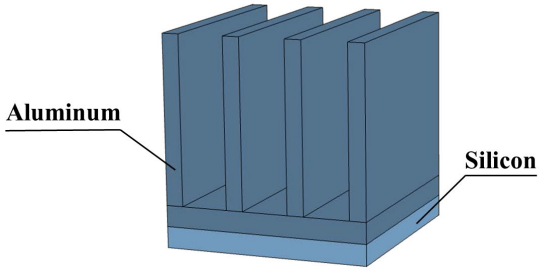

Fig. 5. Profile of the structure for Example II.

TABLE IV
THERMAL PARAMETERS OF THE CHIP

	Silicon	Aluminum
Thermal Conductivity $(\mathrm{W}/[\mathrm{m}\cdot\mathrm{K}])$	131	238
Density $(\mathrm{kg}/\mathrm{m}^3)$	2, 329	2, 700
Heat Capacity $(\mathrm{J}/[\mathrm{kg}\cdot\mathrm{K}])$	700	900

B. Three-Dimensional IC Chip Package

In the second example, a 3-D IC chip package with four pins mounted on a printed circuit board (PCB) is studied. The structure of IC chip package is shown in Fig. 5. The dimensions of the bottom chip, the top chip, and pin are 50 mm × 49 mm × 5 mm, 50 mm × 49 mm × 5 mm, and 50 mm × 4 mm × 40 mm, respectively. The chip is composed of two different materials, whose property details are given in Table IV.

The power distribution of chip is a Gaussian pulse [13] defined by $Q = 10^7 \exp[-t^2/100]$ W/m^3. All the surfaces of the chip are enforced by the convective boundary conditions with an ambient temperature of $T_a = 300$ K and convection coefficient of $h = 200$ W/[m$^2\cdot$K]. The time scale of the example is $t \in [0, 250]$ s. To balance the loss terms, the loss weights are set as $(w_G, w_{BC}, w_E) = (1, 10^6, 10^4)$.

In this example, the performances of four representative methods are compared. Namely, PINN, XPINN, PINN-IC, and the HCD-PINN. As shown in Fig. 5, the structure is composed of six regular rectangular solids. When using methods related to DDM such as XPINN and the HCD-PINN, we teared the solution domain into two small regions based on different materials. However, for the purpose of achieving a uniform distribution of the sampling points, in each rectangular region the number of sampling points is set to be equal. In this example, $N_f = 1536$, $N_b = 768$, and $N_i = 512$. In addition, $N_{\text{inter}} = 512$ sampling points are chosen on the interface between two neighboring subdomains. The details of the comparative experiments are shown as follows.

1) *PINN: Original PINN Method* In this method, the structure of NN is [4] + [60] × 5 + [1].
2) *XPINN: Original XPINN Method* The solution domain is divided into two subdomains. In this method, the structure of NN in each subdomain is [4] + [40] × 5 + [1].
3) *PINN-IC: PINN Method with HC for initial condition*, the details can be seen in Section III-A. In this method, the structure of NN is [4] + [60] × 5 + [1].

TABLE V
COMPARATIVE EXPERIMENTAL RESULT FOR EXAMPLE II

	Mean Absolute Percentage Error (MAPE) of T							Param
	t = 0 s	t = 50 s	t = 100 s	t = 150 s	t = 200 s	t = 250 s	Average	
PINN	1.667%	2.600%	2.016%	1.144%	0.497%	**1.012%**	1.495%	15.0K
XPINN	12.453%	36.314%	17.230%	11.583%	8.940%	3.767%	18.758%	13.6K
PINN-IC	**0.008%**	3.374%	2.911%	0.915%	**0.202%**	2.841%	1.875%	15.0K
HCD-PINN	**0.008%**	**0.704%**	**1.004%**	**0.816%**	0.381%	1.191%	**0.778%**	**11.0K**

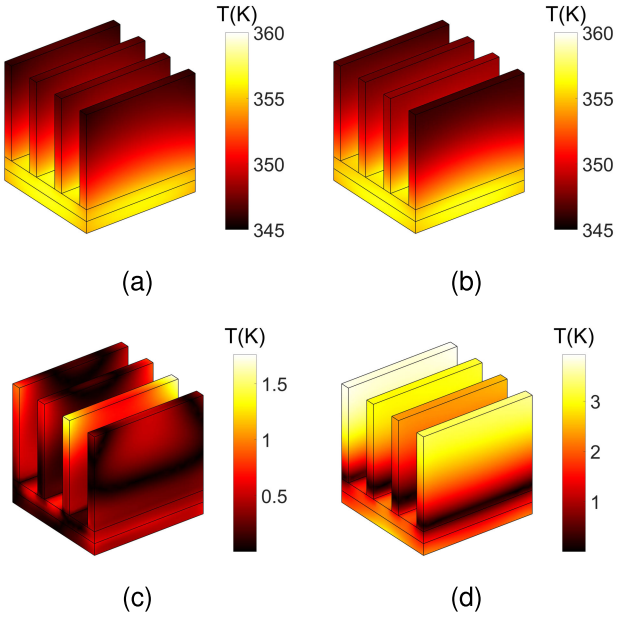

Fig. 6. Temperature distribution (unit: K) at $t = 50$ s by (a) COMSOL and (b) HCD-PINN. Error distribution (unit: K) with respect to COMSOL results at $t = 50$ s by (c) HCD-PINN and (d) PINN.

4) *HCD-PINN: Proposed HCD-PINN* The solution domain is divided into two subdomains. In this method, the structure of NN in this method is NS:[4] + [30] × 5 + [4], and NI:[4] + [30] × 4 + [2].

In this example, all the models are trained using 2000 iterations for Adam optimization and 1000 iterations for L-BFGS optimization. As for the COMSOL simulation, $N_t = 169\,447$ points and 931 874 tetrahedron meshes with second-order Lagrange elements are selected, serving as the reference solution to compare the performance of each model. The computation time of COMSOL is about 386 s. The MAPE at each sampling time, the average MAPE over the entire time, and the sizes of each model are presented in Table V. It can be observed that the proposed method exhibits the highest accuracy in both the almost time and in average. Table VI gives the training and predict time for each method.

In Fig. 6(a) and (b), we compare the temperature distributions solved by COMSOL and the proposed method at $t = 50$ s. In addition, the differences between our proposed method, the PINN method, and the COMSOL simulation are also provided. As can be seen, the maximum temperature error for the proposed method is 1.760 K, while that of PINN is 3.929 K, which shows that our method is more accurate.

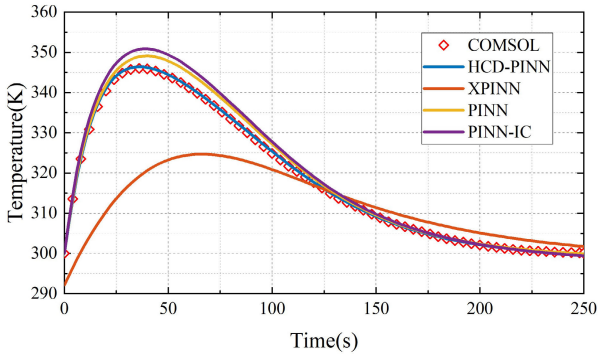

Fig. 7. Transient temperature at $r = (0, 0, 50)$ mm solved by COMSOL, HCD-PINN, xPINN, PINN, and PINN-IC.

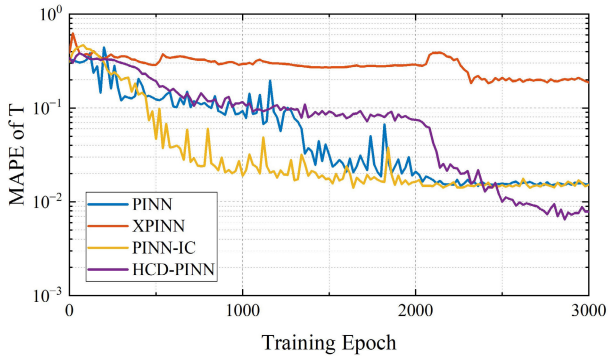

Fig. 8. Trajectories of relative error of T in Example II as a function of training epoch for PINN, XPINN, PINN-IC, and HCD-PINN.

TABLE VI
TRAINING AND PREDICT TIME FOR EXAMPLE II

	Adam (s)	L-BFGS (s)	Training (s)	Prediction (ms)
PINN	0.144	3.110	3398	0.20
XPINN	0.143	2.829	3115	0.17
PINN-IC	0.140	2.509	2789	0.22
HCD-PINN	0.181	3.649	4011	1.02

To further verify the accuracy of the proposed method, in Fig. 7, the transient temperature at $r = (0, 0, 50)$ mm is studied and compared with COMSOL and the other methods. It is evident that the proposed method aligns most closely with

TABLE VII
MAPE OF T ON DIFFERENT ACTIVATION FUNCTIONS FOR EXAMPLE I

	tanh	sigmoid	softplus	swish
PINN	1.912%	**1.705%**	1.670%	1.670%
HCD-PINN	**0.260%**	6.766%	**0.379%**	**0.392%**

TABLE VIII
MAPE OF T ON DIFFERENT ACTIVATION FUNCTIONS FOR EXAMPLE II

	tanh	sigmoid	softplus	swish
PINN	1.495%	4.091%	1.542%	1.440%
HCD-PINN	**0.778%**	**1.732%**	**0.746%**	**0.782%**

TABLE IX
MAPE OF T ON DIFFERENT LOSS WEIGHTS FOR EXAMPLE I

	$w_g = 0.01$	$w_g = 0.1$	$w_g = 1$	$w_g = 10$	$w_g = 100$
$w_b = 0.01$	0.508%	0.492%	0.501%	0.592%	0.960%
$w_b = 0.1$	1.149%	0.383%	0.378%	0.407%	0.473%
$w_b = 1$	1.770%	0.770%	**0.260%**	0.373%	0.385%
$w_b = 10$	6.942%	1.021%	0.490%	0.446%	0.488%
$w_b = 100$	8.414%	75.665%	32.153%	0.502%	0.507%

TABLE X
MAPE OF T ON DIFFERENT LOSS WEIGHTS FOR EXAMPLE II

	$w_g = 0.1$	$w_g = 1$	$w_g = 5$	$w_g = 10$
$w_b = 10^5$	1.113%	7.362%	10.899%	15.460%
$w_b = 10^6$	2.332%	**0.778%**	1.052%	0.968%
$w_b = 5 \times 10^6$	6.381%	1.686%	2.712%	3.512%
$w_b = 10^7$	10.771%	2.183%	4.385%	4.443%

COMSOL simulation results. We plot the variation in MAPE of T during the training epoch in Fig. 8. It can be seen that HCD-PINN has converged to the lowest relative error among different methods.

C. Study of Activation Functions and Loss Weights

In this section, we study the impact of different activation functions and loss weights on the experimental results.

First, we compared the performances of four commonly used activation functions (tanh, sigmoid, softplus, and swish) via two canonical examples. In Tables VII and VIII, the MAPEs of the time-domain temperature T for the two examples with different activation functions are presented. It can be seen that except sigmoid in Example I, HCD-PINN achieves higher accuracy than PINN. Comparing various activation functions, we found that tanh, softplus, and swish yielded similarly excellent results. The poorer accuracy associated with sigmoid stems from its lack of negative output.

Second, to study the influence of unbalance loss terms, we changed the loss weights to take some ablation studies. In each example, we repeated the experiment with fixed w_e but varied w_g, w_b. Tables IX and X give the MAPE of time domain T with different loss weights. In general, different loss weights result in varying accuracies. When loss weights are significantly different in proportion, errors tend to increase. In comparison to Example I, the accuracy in Example II is more affected by loss weights, which is attributed to more complex geometric structure. For Example I, changes in loss weights by around ten times have almost no impact on the error.

V. CONCLUSION

In this article, an HC and DDM enhanced PINN framework is introduced for solving transient thermal problems in complex structures. The application of DDM enhances the representative capability of PINN, allowing the use of multiple neural networks to compute solutions for complex regions. Meanwhile, the HC method ensures continuity between solutions in different subdomains, reducing the number of loss terms that the neural networks need to learn during training. This, in turn, alleviates the computational burden and improves approximation accuracy. Therefore, compared with the traditional PINN and DDM-based XPINN, the proposed method demonstrates higher accuracy in solving transient heat problems involving complex regions with different material compositions. The introduced computational framework empowers PINN to handle complex problem scenarios, paving the way for applications of scientific machine learning.

APPENDIX A

In this Appendix, a general solution for the HC matrix is studied. We take the parameter α_1 as the contrast of the coefficients $\alpha(\boldsymbol{x})$

$$\kappa_1 = \kappa^+(\boldsymbol{x})/\kappa^-(\boldsymbol{x}). \tag{36}$$

The interface condition can be expressed as

$$T^+(\boldsymbol{x}) = T^-(\boldsymbol{x}) \tag{37}$$

$$\kappa_1 \hat{\boldsymbol{n}}^+ \cdot \boldsymbol{p}^+(\boldsymbol{x}) = \hat{\boldsymbol{n}}^- \cdot \boldsymbol{p}^-(\boldsymbol{x}). \tag{38}$$

If dimension of the problem is $d (d \geq 3)$, the HC matrix $\boldsymbol{B}^i(\boldsymbol{x})$ can be defined as $\boldsymbol{B}^i(\boldsymbol{x}) \in \mathbb{R}^{d+1} \times \mathbb{R}^{2\times(d+1)}$. And $\boldsymbol{B}^i$ can be represented using four separate matrices

$$\boldsymbol{B}^+(\boldsymbol{x}) = \begin{bmatrix} 1 & & & \\ & \boldsymbol{N}(\boldsymbol{x}) & \boldsymbol{O} & \boldsymbol{r}(\boldsymbol{x}) \end{bmatrix} \tag{39}$$

$$\boldsymbol{B}^-(\boldsymbol{x}) = \begin{bmatrix} 1 & & & \\ & \boldsymbol{O} & \boldsymbol{N}(\boldsymbol{x}) & \kappa_1 \boldsymbol{r}(\boldsymbol{x}) \end{bmatrix}. \tag{40}$$

The first part is 1 in the first column, which is served for (37), and $\boldsymbol{O}$ is the null matrix with dimension $d \times d$. The vacant positions in the matrices are filled with zeros.

The next part is the basis of the null space for the following equation, in the meanwhile ensuring the degrees of freedom of the NN outputs:

$$\boldsymbol{n}(\boldsymbol{x}) \cdot \boldsymbol{p}(\boldsymbol{x}) = 0. \tag{41}$$

The matrix $\boldsymbol{N}(\boldsymbol{x}) \in \mathbb{R}^d \times \mathbb{R}^d$ is given by

$$\boldsymbol{N}(\boldsymbol{x}) = [\beta_1(\boldsymbol{x}); \beta_2(\boldsymbol{x}); \ldots; \beta_d(\boldsymbol{x})]^{\mathrm{T}}. \tag{42}$$

Here, $\beta_k(\boldsymbol{x})$ is the Gram–Schmidt orthogonalization [34] of normal vector $\hat{\boldsymbol{n}}(\boldsymbol{x})$, which is performed as

$$\beta_k(\boldsymbol{x}) = \boldsymbol{e}_k - \big(\boldsymbol{e}_k \cdot \hat{\boldsymbol{n}}(\boldsymbol{x})\big)\hat{\boldsymbol{n}}(\boldsymbol{x}), \quad k = 1, \ldots, d \tag{43}$$

where $\boldsymbol{e}_k$ is the kth row of the identity matrix $\boldsymbol{I}_d$. The matrix $\boldsymbol{N}(\boldsymbol{x})$ can be expressed as

$$\boldsymbol{N}(\boldsymbol{x}) = \boldsymbol{I}_d - \hat{\boldsymbol{n}}(\boldsymbol{x})^{\mathrm{T}}\hat{\boldsymbol{n}}(\boldsymbol{x}) \tag{44}$$

which satisfies that

$$\begin{aligned}\hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \boldsymbol{N}(\boldsymbol{x})\boldsymbol{R} &= \hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \big(\boldsymbol{I}_d - \hat{\boldsymbol{n}}(\boldsymbol{x})^{\mathrm{T}}\hat{\boldsymbol{n}}(\boldsymbol{x})\big)\boldsymbol{R} \\ &= \big(\hat{\boldsymbol{n}}(\boldsymbol{x}) - \hat{\boldsymbol{n}}(\boldsymbol{x})\hat{\boldsymbol{n}}(\boldsymbol{x})^{\mathrm{T}}\hat{\boldsymbol{n}}(\boldsymbol{x})\big)\boldsymbol{R} \\ &= 0 \end{aligned} \tag{45}$$

where $\boldsymbol{R} \in \mathbb{R}^d$ is an arbitrary column vector.

The last part is a column vector $\boldsymbol{r}(x)$, whose dimension is d

$$\boldsymbol{r}(\boldsymbol{x}) = [n_1(\boldsymbol{x}), n_2(\boldsymbol{x}), \ldots, n_d(\boldsymbol{x})]^{\mathrm{T}}. \tag{46}$$

The proof of the proposed HC matrix is expressed as follows. The unknowns $(u^+(\boldsymbol{x}), \boldsymbol{p}^+(\boldsymbol{x}))$ and $(u^-(\boldsymbol{x}), \boldsymbol{p}^-(\boldsymbol{x}))$ in the interface are transformed as

$$(u^+(\boldsymbol{x}), \boldsymbol{p}^+(\boldsymbol{x})) = \boldsymbol{B}^+(\boldsymbol{x})\mathrm{NI}^j(\boldsymbol{x}) \tag{47}$$

$$(u^-(\boldsymbol{x}), \boldsymbol{p}^-(\boldsymbol{x})) = \boldsymbol{B}^-(\boldsymbol{x})\mathrm{NI}^j(\boldsymbol{x}). \tag{48}$$

After decomposing the HC matrices, we obtain the following expressions:

$$u^+(\boldsymbol{x}) = \mathrm{NI}^j(\boldsymbol{x})[1] = u^-(\boldsymbol{x}) \tag{49}$$

where $\mathrm{NI}^j(\boldsymbol{x})[1]$ means the first column of output of $\mathrm{NI}^j(\boldsymbol{x})$, and (37) is satisfied.

By incorporating (45) and (46) into (47) and (48), we get

$$\begin{aligned}\hat{\boldsymbol{n}}^+(\boldsymbol{x}) \cdot \boldsymbol{p}^+(\boldsymbol{x}) &= \hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \boldsymbol{N}(x)\mathrm{NI}^j(\boldsymbol{x})[2 : \mathrm{d}+1] \\ &\quad + \hat{\boldsymbol{n}} \cdot \boldsymbol{r}(\boldsymbol{x})\mathrm{NI}^j(\boldsymbol{x})[2\mathrm{d}+2] \\ &= \hat{\boldsymbol{n}} \cdot \boldsymbol{r}(\boldsymbol{x})\mathrm{NI}^j(\boldsymbol{x})[2\mathrm{d}+2] \end{aligned} \tag{50}$$

$$\begin{aligned}\hat{\boldsymbol{n}}^-(\boldsymbol{x}) \cdot \boldsymbol{p}^-(\boldsymbol{x}) &= \hat{\boldsymbol{n}}(\boldsymbol{x}) \cdot \boldsymbol{N}(x)\mathrm{NI}^j(\boldsymbol{x})[\mathrm{d}+2 : 2\mathrm{d}+1] \\ &\quad + \kappa_1\hat{\boldsymbol{n}} \cdot \boldsymbol{r}(\boldsymbol{x})\mathrm{NI}^j(\boldsymbol{x})[2\mathrm{d}+2] \\ &= \kappa_1\hat{\boldsymbol{n}} \cdot \boldsymbol{r}(\boldsymbol{x})\mathrm{NI}^j(\boldsymbol{x})[2\mathrm{d}+2]. \end{aligned} \tag{51}$$

From (50) and (51), the interface condition (38) is satisfied automatically. And the proposed HC matrix (39) and (40) satisfies the interface condition. In addition, $\boldsymbol{p}^-(\boldsymbol{x})$ and $\boldsymbol{p}^+(\boldsymbol{x})$ are composed of different parts of $\mathrm{NI}^j(\boldsymbol{x})$. Therefore, they are linearly independent, ensuring the degrees of freedom of the outputs.

To validate the proposed HC matrix, a 3-D normal vector example is studied. Similar to Section II, the solution domain is composed of two geometries. One is a cube with size of $1.5 \times 1.5 \times 1.5$ labeled as Region I, and the other is a sphere with a radius of 1 labeled as Region II. The material property details of the region are shown in Table I. Other parameters of the problem are set as follows: $Q = 50\ \mathrm{W/m^3}$, $h = 1\ \mathrm{W/[m^2 \cdot K]}$, time scale $t \in [0, 4]$ s, $T_0 = 300$ K, $T_a = 300$ K, $\mathrm{NN_{main}}$:[4] + [40] × 5 + [4], and $\mathrm{NN_{inter}}$:[4] + [30] × 4 + [8]. The framework is trained with 3000 iterations of Adam optimization and 1000 iterations of L-BFGS optimization.

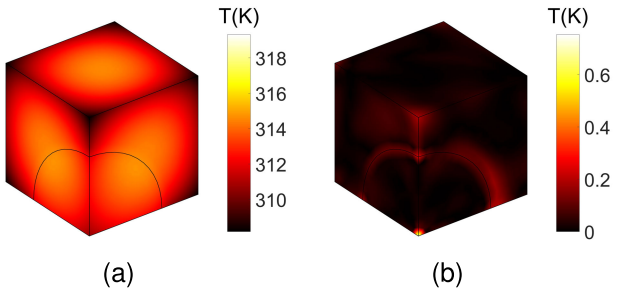

Fig. 9. Temperature distribution (unit: K) at $t = 4$ s by (a) COMSOL. Error distribution (unit: K) with respect to the exact solution at $t = 4$ s by (b) HCD-PINN.

TABLE XI
COMPARATIVE EXPERIMENTAL RESULT FOR DISTANCE FUNCTIONS

	Average MAE of T	
	Example I	Example II
HCD-PINN	**0.260%**	**0.778%**
HC-Net	0.484%	2.146%

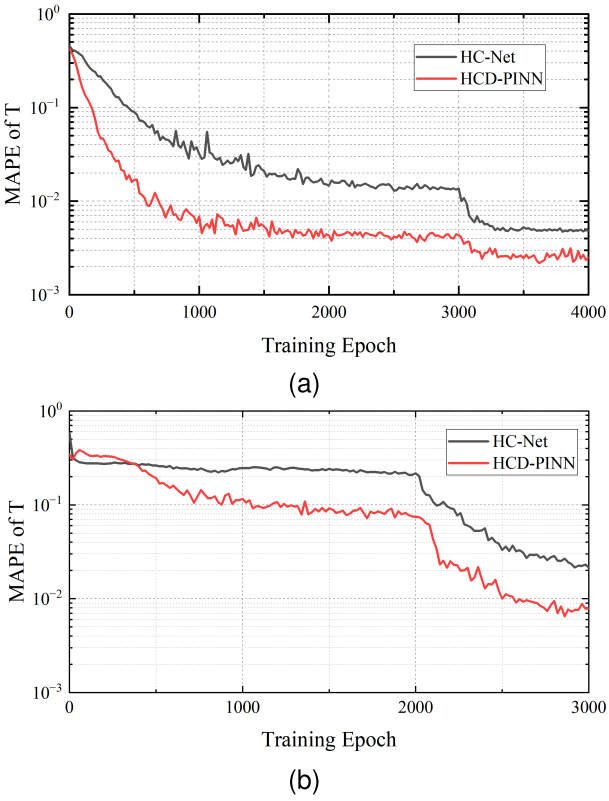

Fig. 10. Variation in relative error compared with HC-Net and HCD-PINN for (a) Example I and (b) Example II.

The temperature distribution by COMSOL and error distribution comparing the proposed method and COMSOL is shown in Fig. 9. The maximum temperature error compared with COMSOL is 0.7529 K. The mean relative error in total time is 0.405%. The experimental result shows the efficiency of the proposed HC matrix.

APPENDIX B

In this appendix, we compare the proposed distance function with that shown in [34]. HC-net is a unified framework for geometrically complex PDEs by introducing HC for BCs and ICs. In HC-net, the two types of distance functions are $l^q(\cdot)$ and $\exp[-\beta_s l^q(\cdot)]$, corresponding to $\mathcal{D}_1(\cdot)$ and $\mathcal{D}_2(\cdot)$ in this article, respectively. $\beta_s \in \mathbb{R}$ is a hyperparameter of the "hardness" in the spatial domain. The inputs of distance functions are the normalized distance between the points $\boldsymbol{x}$ and interfaces.

To validate that the proposed distance functions, $\mathcal{D}_1(\cdot)$ and $\mathcal{D}_2(\cdot)$ are more suitable for the interface problems. We take comparison experiments presented in Section IV with two kinds of distance functions. In the above experiments, all the network structures and domain decomposition results are identical. The hyperparameter β in (28) and (29) is set to 0.01, and β_s in HC-Net is set to 5.

The average MAE results of both the methods are shown in Table XI. The variations in relative errors during training are studied in Fig. 10. From the experimental results, it is evident that compared with the distance function proposed by HC-Net, the distance function used in our proposed method achieves higher accuracy. Moreover, during the training process, our method exhibits faster convergence and is more straightforward to train.

REFERENCES

[1] R. R. Tummala, "SOP: What is it and why? A new microsystem-integration technology paradigm-Moore's law for system integration of miniaturized convergent systems of the next decade," *IEEE Trans. Adv. Packag.*, vol. 27, no. 2, pp. 241–249, May 2004.

[2] J. U. Knickerbocker et al., "3-D silicon integration and silicon packaging technology using silicon through-vias," *IEEE J. Solid-State Circuits*, vol. 41, no. 8, pp. 1718–1725, Aug. 2006.

[3] S. Rzepka, K. Banerjee, E. Meusel, and C. Hu, "Characterization of self-heating in advanced VLSI interconnect lines based on thermal finite element simulation," *IEEE Trans. Compon., Packag., Manuf. Technol., A*, vol. 21, no. 3, pp. 406–411, Sep. 1998.

[4] Y.-B. Shi, W.-Y. Yin, J.-F. Mao, P. Liu, and Q. H. Liu, "Transient electrothermal analysis of multilevel interconnects in the presence of ESD pulses using the nonlinear time-domain finite-element method," *IEEE Trans. Electromagn. Compat.*, vol. 51, no. 3, pp. 774–783, Sep. 2009.

[5] T. Lu and J. M. Jin, "Transient electrical-thermal analysis of 3-D power distribution network with FETI-enabled parallel computing," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 4, no. 10, pp. 1684–1695, Oct. 2014.

[6] J. Chung and G. M. Hulbert, "A time integration algorithm for structural dynamics with improved numerical dissipation: The Generalized-α method," *J. Appl. Mech.*, vol. 60, no. 2, pp. 371–375, Jun. 1993.

[7] S. Dosopoulos and J.-F. Lee, "Interior penalty discontinuous Galerkin finite element method for the time-dependent first order Maxwell's equations," *IEEE Trans. Antennas Propag.*, vol. 58, no. 12, pp. 4085–4090, Dec. 2010.

[8] J.-P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *J. Comput. Phys.*, vol. 114, no. 2, pp. 185–200, Oct. 1994.

[9] A. Cheng, S. Chen, H. Zeng, D. Ding, and R. Chen, "Transient analysis for electrothermal properties in nanoscale transistors," *IEEE Trans. Electron Devices*, vol. 65, no. 9, pp. 3930–3935, Sep. 2018.

[10] A. Fournier, H.-P. Bunge, R. Hollerbach, and J.-P. Vilotte, "A Fourier-spectral element algorithm for thermal convection in rotating axisymmetric containers," *J. Comput. Phys.*, vol. 204, no. 2, pp. 462–489, Apr. 2005.

[11] Q. Ren, L. E. Tobón, Q. Sun, and Q. H. Liu, "A new 3-D nonspurious discontinuous Galerkin spectral element time-domain (DG-SETD) method for Maxwell's equations," *IEEE Trans. Antennas Propag.*, vol. 63, no. 6, pp. 2585–2594, Jun. 2015.

[12] P. Li, Y. Dong, M. Tang, J. Mao, L. J. Jiang, and H. Bağci, "Transient thermal analysis of 3-D integrated circuits packages by the DGTD method," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 7, no. 6, pp. 862–871, Jun. 2017.

[13] Y. L. Dong, M. Tang, P. Li, and J. F. Mao, "Transient electromagnetic-thermal simulation of dispersive media using DGTD method," *IEEE Trans. Electromagn. Compat.*, vol. 61, no. 4, pp. 1305–1313, Aug. 2019.

[14] A. F. Yang, M. Tang, J. F. Mao, L. J. Jiang, H. Bagcii, and P. Li, "DC IR-drop analysis of power distribution networks by a Robin transmission condition-enhanced discontinuous Galerkin method," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 12, no. 1, pp. 89–99, Jan. 2022.

[15] X. Zhang, P. Li, X. C. Li, L. J. Jiang, and J. F. Mao, "A hybridizable discontinuous Galerkin time-domain method with Robin transmission condition for transient thermal analysis of 3-D integrated circuits," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 12, no. 9, pp. 1474–1483, Sep. 2022.

[16] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods*. Berlin, Germany: Springer, 2008.

[17] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.

[18] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Rev.*, vol. 63, no. 1, pp. 208–228, Feb. 2021.

[19] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, pp. 1–43, Apr. 2018.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[21] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, nos. 1–3, pp. 503–528, Aug. 1989.

[22] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Comput. Methods Appl. Mech. Eng.*, vol. 360, Mar. 2020, Art. no. 112789.

[23] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, Dec. 2021.

[24] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations," *J. Comput. Phys.*, vol. 426, Feb. 2021, Art. no. 109951.

[25] Y. Chen, L. Lu, G. E. Karniadakis, and L. D. Negro, "Physics-informed neural networks for inverse problems in nano-optics and metamaterials," *Opt. Exp.*, vol. 28, no. 8, pp. 11618–11633, Apr. 2020.

[26] D. Liu and Y. Wang, "Multi-fidelity physics-constrained neural network and its application in materials modeling," *J. Mech. Design*, vol. 141, no. 12, Dec. 2019, Art. no. 121403.

[27] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," *J. Heat Transf.*, vol. 143, no. 6, Apr. 2021, Art. no. 060801.

[28] R. Laubscher, "Simulation of multi-species flow and heat transfer using physics-informed neural networks," *Phys. Fluids*, vol. 33, no. 8, Aug. 2021, Art. no. 087101.

[29] Q. Zheng, L. Zeng, and G. E. Karniadakis, "Physics-informed semantic inpainting: Application to geostatistical modeling," *J. Comput. Phys.*, vol. 419, Oct. 2020, Art. no. 109676.

[30] C. Song, T. Alkhalifah, and U. B. Waheed, "A versatile framework to solve the Helmholtz equation using physics-informed neural networks," *Geophys. J. Int.*, vol. 228, no. 3, pp. 1750–1762, Nov. 2021.

[31] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Comput. Methods Appl. Mech. Eng.*, vol. 365, Jun. 2020, Art. no. 113028.

[32] A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," *Commun. Comput. Phys.*, vol. 28, no. 5, pp. 2002–2041, Jun. 2020.

[33] B. Moseley, A. Markham, and T. Nissen-Meyer, "Finite basis physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations," *Adv. Comput. Math.*, vol. 49, no. 4, Jul. 2023, Art. no. 62.

[34] S. Liu, Z. Hao, C. Ying, H. Su, J. Zhu, and Z. Cheng, "A unified hard-constraint framework for solving geometrically complex pdes," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, 2022, pp. 20287–20299.

[35] H. Sheng and C. Yang, "PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries," *J. Comput. Phys.*, vol. 428, Mar. 2021, Art. no. 110085.

[36] J. Berg and K. Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries," *Neurocomputing*, vol. 317, pp. 28–41, Nov. 2018.

[37] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM J. Sci. Comput.*, vol. 43, no. 6, pp. B1105–B1132, Jan. 2021.

[38] L. Lyu, Z. Zhang, M. Chen, and J. Chen, "MIM: A deep mixed residual method for solving high-order partial differential equations," *J. Comput. Phys.*, vol. 452, Mar. 2022, Art. no. 110930.

[39] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 34, 2021, pp. 26548–26560.

[40] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, "PPINN: Parareal physics-informed neural network for time-dependent PDEs," *Comput. Methods Appl. Mech. Eng.*, vol. 370, Oct. 2020, Art. no. 113250.

[41] L. Yuan, Y.-Q. Ni, X.-Y. Deng, and S. Hao, "A-PINN: A auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations," *J. Comput. Phys.*, vol. 462, p. 111260, Aug. 2022.

[42] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019, pp. 8024–8035.

[43] F. Cao, X. Guo, F. Gao, and D. Yuan, "Deep learning nonhomogeneous elliptic interface problems by soft constraint physics-informed neural networks," *Mathematics*, vol. 11, no. 8, p. 1843, Apr. 2023.

[44] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, "A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks," *Comput. Methods Appl. Mech. Eng.*, vol. 403, Jan. 2023, Art. no. 115671.