Journal of Computational Physics 530 (2025) 113906

Contents lists available at ScienceDirect

Journal of Computational Physics

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

Error analysis and numerical algorithm for PDE approximation with hidden-layer concatenated physics informed neural networks

Yanxia Qian [a], Yongchao Zhang [b], Suchuan Dong [c,*]

[a] Department of Mathematics, School of Science, Xi'an University of Technology, Xi'an, Shaanxi, 710048, China
[b] School of Mathematics, Northwest University, Xi'an, Shaanxi 710069, China
[c] Center for Computational and Applied Mathematics, Department of Mathematics, Purdue University, West Lafayette, IN47907, USA

ARTICLE INFO

Keywords:
Physics informed neural network
Hidden-layer concatenation
Block time-marching
Deep neural networks
Deep learning
Scientific machine learning

ABSTRACT

We present the hidden-layer concatenated physics informed neural network (HLConcPINN) method, which combines hidden-layer concatenated feed-forward neural networks, a modified block time marching strategy, and a physics informed approach for approximating partial differential equations (PDEs). We analyze the convergence properties and establish the error bounds of this method for two types of PDEs: parabolic (exemplified by the heat and Burgers' equations) and hyperbolic (exemplified by the wave and nonlinear Klein-Gordon equations). We show that its approximation error of the solution can be effectively controlled by the training loss for dynamic simulations with long time horizons. The HLConcPINN method in principle allows an arbitrary number of hidden layers not smaller than two and any of the commonly-used smooth activation functions for the hidden layers beyond the first two, with theoretical guarantees. This generalizes several recent neural network techniques, which have theoretical guarantees but are confined to two hidden layers in the network architecture and the tanh activation function. Our theoretical analyses subsequently inform the formulation of appropriate training loss functions for these PDEs, leading to physics informed neural network (PINN) type computational algorithms that differ from the standard PINN formulation. Ample numerical experiments are presented based on the proposed algorithm to validate the effectiveness of this method and confirm aspects of the theoretical analyses.

1. Introduction

The rapid growth in data availability and computing resources has ushered in a transformative era in machine learning and data analytics, fueling remarkable advancements across diverse scientific and engineering disciplines [33]. These breakthroughs have a significant impact on fields such as natural language processing, robotics, computer vision, speech and image recognition, and genomics. Of particular promise is the use of neural network (NN) based approaches to tackle challenges such as high-dimensional problems, including high-dimensional partial differential equation (PDE). This is due to the intractable computational workload caused by the curse of dimensionality associated with conventional numerical techniques, rendering such techniques practically infeasible. Deep learning algorithms, on the other hand, can offer invaluable support. Pertaining to PDE problems specifically, neural

* Corresponding author.
E-mail addresses: yxqian@xaut.edu.cn (Y. Qian), yoczhang@nwu.edu.cn (Y. Zhang), sdong@purdue.edu (S. Dong).

https://doi.org/10.1016/j.jcp.2025.113906
Received 10 June 2024; Received in revised form 27 February 2025; Accepted 1 March 2025

Available online 7 March 2025

network methods provide implicit regularization, exhibiting a great potential to alleviate or overcome challenges related to high dimensionality [3,4].

This surge of progress has driven extensive research efforts in recent years, fostering the integration of deep learning techniques into scientific computing [31,46,42,20,34]. Notably, the physics informed neural networks (PINNs) approach, introduced in [42], has demonstrated remarkable success in addressing various forward and inverse PDE problems, establishing itself as a widely adopted methodology in scientific machine learning [42,25,10,30,52,29,7,47,16,8,49,22,32,18,19,50,17,45,27,28,40]. Comprehensive reviews of PINNs, including their benefits and limitations, can be found in [31,9].

Theoretical understanding of the physics informed neural network approach has attracted extensive research, and contributions to the theoretical analysis of PDEs using PINNs have grown steadily and substantially in recent years. Shin et al. [43,44] conducted an extensive investigation into PINNs, demonstrating their consistency when applied to linear elliptic and parabolic PDEs. Mishra and Molinaro proposed an abstract framework to estimate the generalization error of PINNs in forward PDE problems [37] and extended it to inverse PDE problems in [36]. Bai and Koley [1] focused on evaluating the approximation performance of PINNs in nonlinear dispersive PDEs. Biswa et al. [6] supplied error estimates and stability analysis for the incompressible Navier-Stokes equations. Zerbinati [53] treated PINNs as a point matching localization method and provided error estimates for elliptic problems. De Ryck et al. [13] presented crucial theoretical findings on PINNs with tanh activation functions and analyzed their approximation errors. These results underlie the theoretical studies on PINNs for the Navier-Stokes equations [12], high-dimensional radiative transfer problem [35], and dynamic PDEs of second order in time [41]. Hu et al. [26] provided valuable insights into the accuracy and convergence properties of PINNs for approximating the primitive equations. Berrone et al. [5] conducted a posteriori error analysis of variational PINNs for solving elliptic boundary-value problems. In [27] the generalized Barron space has been considered for the neural network and a priori and posteriori generalization bound on the PDE residuals are provided. Pantidis et al. [39] concentrated on two critical aspects: error convergence and engineering-guided hyperparameter search, aiming to optimize the performance of the integrated finite element neural network. Gao and Zakharian [24] shed insight into the error estimation for solving nonlinear equations using PINNs in the context of $\mathbb{R}$-smooth Banach spaces.

The analyses of PINNs in the aforementioned contributions all involve feed-forward neural networks (FNNs). The network output represents the PDE solution, and the neural network is trained by a "physics informed" approach, i.e. by minimizing a loss function related to residuals of the PDE and the boundary/initial conditions. The PINN methods of [12,35,41] (among others) theoretically guarantee for a class of PDEs that (i) the approximation error of the solution field will be bounded by the training loss, and (ii) there indeed exist FNNs that can make the training loss arbitrarily small. While these methods [12,35,41] with theoretical guarantees are attractive and important, they suffer from two limitations: (i) the neural network must have two hidden layers, and (ii) the activation function is restricted to tanh (hyperbolic tangent) only. These restrictions stem from the theoretical result about tanh neural networks of [12], which has been used to establish the findings in these studies.

We are interested in the following question:

- Is it possible to develop a PINN technique that retains these theoretical guarantees and additionally allows (i) an arbitrary number of hidden layers larger than two, and (ii) activation functions other than tanh?

In this work we develop a PINN approach to address the above question in the context stemming from the theoretical result of [12]. Our method provides an answer in the affirmative. It relies on the theoretical result of [12], but alleviates and largely overcomes the two aforementioned limitations. We would like to point out that activation functions other than tanh have long been used in practice, which is obviously not new. What is new lies in that, with the approach stemming from the theoretical finding of [12], the method presented herein removes those restrictions and allows the use of neural networks with more than two hidden layers and activation functions other than tanh, while preserving the theoretical guarantees.

A key strategy in our approach is the adoption of a type of modified FNNs, known as hidden-layer concatenated feed-forward neural networks (HLConcFNNs). HLConcFNN was proposed by [38] originally for extreme learning machines (ELMs), in order to overcome the issue that achieving high accuracy often necessitates a wide last hidden layer in conventional ELMs [14,19,17,51]. Building upon FNNs, HLConcFNNs establish direct connections between all hidden layer nodes and the output layer through a logical concatenation of the hidden layers. HLConcFNNs have the interesting property that, by appending hidden layers or adding extra nodes to existing hidden layers of a network structure, its representation capacity is guaranteed to be not decreasing (in practice strictly increasing with nonlinear activation functions) [38]. By contrast, conventional FNNs lack such a property. The properties of HLConcFNNs prove crucial to generalizing the theoretical analysis of PINN to network architectures with more than two hidden layers and with other activation functions than tanh. We would like to point out that the hidden-layer concatenated extreme learning machines (HLConcELMs) developed in [38] employ HLConcFNNs as the base architecture and have all the hidden-layer parameters randomly assigned and fixed (non-trainable), leaving only those parameters associated with the direct connections between the hidden-layer nodes and the output nodes trainable. On the other hand, the theoretical analyses presented in this work rely on HLConcFNNs in which all the network parameters (including those in the hidden layers) are trainable.

Another strategy in our approach and theoretical analysis is the block time marching (BTM) scheme [14] for dynamic simulations of time-dependent PDEs with long (or longer) time horizons. For long-time dynamic simulations, training the neural network on the entire spatial-temporal domain with a large dimension in time proves to be especially difficult. In this case, dividing the domain into "time blocks" with moderate sizes and training the neural network on the space-time domain of each time block individually and successively, with the initial conditions informed by computation results from the preceding time block, can significantly improve the accuracy and ease the training [14]. This is the essence of block time marching. We refer to e.g. [32,23,40] (among others)

for analogous strategies. Block time marching, as formulated in its existing form [14], is not amenable to theoretical analysis. The problem lies in the data regularity for the initial conditions, when multiple time blocks are present. To overcome this issue, we present in this work a modified BTM scheme, denoted by "ExBTM" (standing for Extended BTM), which enables the analysis of the block time marching strategy.

In this paper we present the hidden-layer concatenated PINN (or HLConcPINN) method, by combining hidden-layer concatenated FNNs, the modified block time marching strategy, and the physics informed neural network approach, for approximating parabolic and hyperbolic type PDEs. We analyze the convergence properties and error bounds of this method for parabolic equations, exemplified by the heat and viscous Burgers' equations, and hyperbolic equations, exemplified by the wave and nonlinear Klein-Gordon equations. In addition, we also analyze this method, excluding the block time marching component, for the elliptic type equation, exemplified by the nonlinear Helmholtz equation. Our analyses show that the approximation error of the HLConcPINN solution can be effectively controlled by the training error for long-term dynamic simulations. The network architecture for HLConcPINNs can in principle contain any number of hidden layers larger than two, and the activation function for all hidden layers beyond the first two can essentially be any of the commonly-used activation functions with sufficient regularity, as long as the first two hidden layers adopt the tanh activation function.

These theoretical analyses subsequently inform the formulation of appropriate training loss functions, giving rise to PINN-type computational algorithms, which differ from the standard PINN and BTM formulations for these PDEs. We present ample numerical experiments based on the proposed algorithm. The numerical results demonstrate the effectiveness of this method in accurately capturing the solution field and affirm the relationship between the approximation error and the training loss from the theoretical analyses. Extensive numerical comparisons between the current algorithm and that employing the original BTM scheme are also presented.

The main contributions of this paper lie in two aspects: (i) the hidden-layer concatenated PINN methodology, and (ii) the analyses of the convergence properties and error estimates for this technique. The HLConcPINN method has the salient property that it allows an arbitrary number of hidden layers not smaller than two in the network structure, and allows essentially all of the commonly-used smooth activation functions, with theoretical guarantees.

The remainder of this paper is structured as follows. Section 2 provides an overview of HLConcPINN and the BTM strategy. In Sections 3–6, we analyze the convergence and errors of the HLConcPINN algorithm for approximating the heat equation, Burgers' equation, wave equation and the nonlinear Klein-Gordon equation. Section 7 provides a set of numerical experiments with these PDEs to show the effectiveness of the HLConcPINN method and to supplement our theoretical analyses. Section 8 concludes the presentation with some further remarks. Finally, the appendix (Section 9) summarizes several auxiliary results and provides proofs for the theorems discussed in Section 3. In addition, the appendix provides an analysis of the HLConcPINN method for approximating the convection equation, as well as this method (without the block time marching component) for the nonlinear Helmholtz equation.

2. Hidden-layer concatenated physics informed neural networks and block time marching

2.1. Generic PDE

Consider a compact domain $D \subset \mathbb{R}^d$ ($d > 0$ being an integer) and the following initial/boundary value problem on this domain,

$$\frac{\partial u}{\partial t}(\boldsymbol{x},t) + \mathcal{L}[u](\boldsymbol{x},t) = 0 \qquad (\boldsymbol{x},t) \in D \times [0,T], \tag{1a}$$

$$\mathcal{B}u(\boldsymbol{x},t) = u_d(\boldsymbol{x},t) \qquad (\boldsymbol{x},t) \in \partial D \times [0,T], \tag{1b}$$

$$u(\boldsymbol{x},0) = u_{in}(\boldsymbol{x}) \qquad \boldsymbol{x} \in D, \tag{1c}$$

Here, $u : D \times [0,T] \subset \mathbb{R}^{d+1} \to \mathbb{R}^m$ ($m \geq 1$ being an integer) is the unknown field solution. u_d is the boundary data, and u_{in} is the initial distribution for u. $\mathcal{L}$ and $\mathcal{B}$ denote the differential and boundary operators. T is the dimension in time.

2.2. Physics informed neural networks

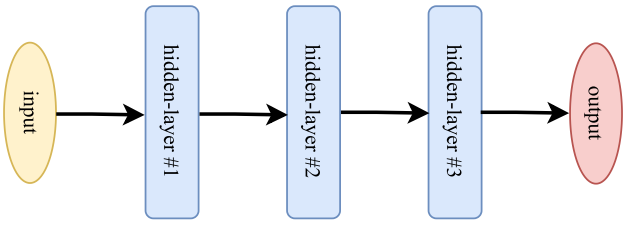
Physics informed neural network (PINN) refers to the general approach for approximating a PDE problem using feedforward neural networks (FNNs) by minimizing the residuals involved in the problem. We first define feedforward neural networks, and then discuss the related machinery for the analysis of PINN.

Let $\sigma : \mathbb{R} \to \mathbb{R}$ denote an activation function. For any $n \in \mathbb{N}$ and $z = (z_1, z_2, \cdots, z_n) \in \mathbb{R}^n$, we define $\sigma(z) := (\sigma(z_1), \cdots, \sigma(z_n))$. A feedforward neural network (with three hidden layers) is illustrated in Fig. 1(a) using a cartoon. It is formally defined as follows.

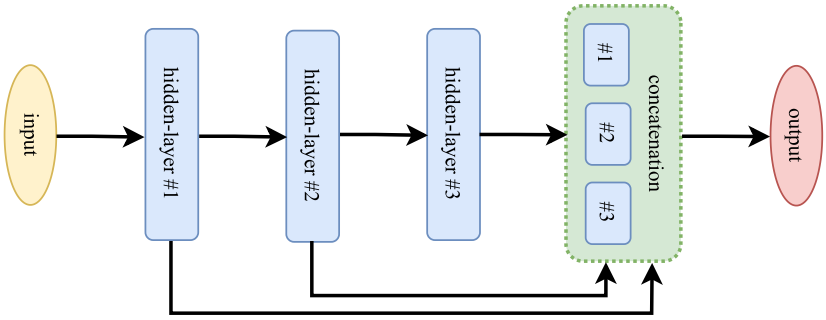
Definition 2.1. Let W and L be integers, and $1 \leq l_i \leq W$ ($0 \leqslant i \leqslant L$) denote $(L+1)$ positive integers. Let $R \in \mathbb{R}$ represent a bounded positive real number, $z_0 \in \mathbb{R}^{l_0}$ denote the input variable, $\sigma : \mathbb{R} \to \mathbb{R}$ be a twice differentiable activation function, and $\mathcal{A}_k^{\vartheta}$ ($1 \leq k \leq L$) be an affine mapping $\mathcal{A}_k^{\vartheta} : \mathbb{R}^{l_{k-1}} \to \mathbb{R}^{l_k}$ given by

$$z_{k-1} \mapsto W_k z_{k-1} + b_k \qquad \text{for } 1 \leq k \leq L,$$

where $W_k \in [-R,R]^{l_k \times l_{k-1}} \subset \mathbb{R}^{l_k \times l_{k-1}}$ and $b_k \in [-R,R]^{l_k} \subset \mathbb{R}^{l_k}$ are referred to as the weight/bias coefficients for $1 \leq k \leq L$. Let Θ denote the collection of all weights/biases and $\vartheta \in \Theta$.



(a) Conventional FNN



(b) Hidden-layer concatenated FNN

Fig. 1. Illustration of network structures (with 3 hidden layers) for conventional and hidden-layer concatenated neural networks. In hidden-layer concatenated FNN, all the hidden nodes are exposed to the output nodes, while in conventional FNN only the last hidden-layer nodes are exposed to the output nodes.

A feedforward neural network is defined as the mapping $u_\vartheta : \mathbb{R}^{l_0} \to \mathbb{R}^{l_L}$ given by

$$u_\vartheta(z_0) = \mathcal{A}_L^\vartheta \circ \sigma \circ \mathcal{A}_{L-1}^\vartheta \circ \cdots \circ \sigma \circ \mathcal{A}_1^\vartheta(z_0) \qquad z_0 \in \mathbb{R}^{l_0}, \tag{2}$$

where $\circ$ denotes a function composition.

The feedforward neural network in the definition contains $(L+1)$ layers ($L \geq 2$) with widths $(l_0, l_1, \cdots, l_L)$, respectively. The input layer and the output layer have l_0 and l_L nodes, respectively. The $(L-1)$ layers between the input/output layers are the hidden layers, with widths l_k $(1 \leq k \leq L-1)$. From layer to layer, the network logic represents an affine transform, followed by a function composition with the activation function σ. No activation function is applied to the output layer. Hereafter we refer to the vector of positive integers, $\boldsymbol{l} = (l_0, l_1, \ldots, l_L)$, as an architectural vector, which characterizes the architecture of an FNN.

The neural network u_ϑ, defined by (2), is a parameterized function of the input $z_0 = (\boldsymbol{x}, t)$, with the parameter ϑ of weights and biases. We represent the solution field u to problem (1) by the neural network u_ϑ, and wish to find the parameters ϑ such that u_ϑ approximates u well.

It is necessary to approximate function integrals during the analysis of physics informed neural networks. Given a subset $\Lambda \subset \mathbb{R}^d$ and a function $f \in L^1(\Lambda)$, a quadrature rule provides an approximation of the integral by $\int_\Lambda f(z)\mathrm{d}z \approx \frac{1}{M}\sum_{n=1}^M \omega_n f(z_n)$, where $z_n \in \Lambda$ $(1 \leq n \leq M)$ represents the quadrature points and ω_n $(1 \leq n \leq M)$ denotes the appropriate quadrature weights. The approximation accuracy is influenced by the regularity of f, the type of quadrature rule and the number of quadrature points (M). In the partial differential equations considered in this work, we assume that the problem dimension is low, thus allowing the use of standard deterministic values for the integrating points. Following [12,41], we employ the midpoint rule for numerical integrals. We partition Λ into $M \sim N^d$ cubes with an edge length $\frac{1}{N}$. The approximation accuracy is determined by

$$\left| \int_\Lambda f(z)\mathrm{d}z - \mathcal{Q}_M^\Lambda[f] \right| \leq C_f M^{-2/d}, \tag{3}$$

where $\mathcal{Q}_M^\Lambda[f] := \frac{1}{M}\sum_{n=1}^M f(z_n)$, $C_f \lesssim \|f\|_{C^2(\Lambda)}$ ($a \lesssim b$ denotes $a \leq Cb$ for some constant C) and $\{z_n\}_{n=1}^M$ denote the midpoints of these cubes [11].

2.3. Hidden-layer concatenated physics informed neural networks (HLConcPINNs)

We consider a type of modified FNN, termed hidden-layer concatenated feed-forward neural network (HLConcFNN) proposed by [38], for the PDE approximation in this work. HLConcFNNs differ from traditional FNNs by a modification that establishes direct connections between all hidden nodes and the output layer. The modified network, as illustrated in Fig. 1(b) with three hidden layers, incorporates a logical concatenation layer between the last hidden layer and the output layer. This layer aggregates the output fields

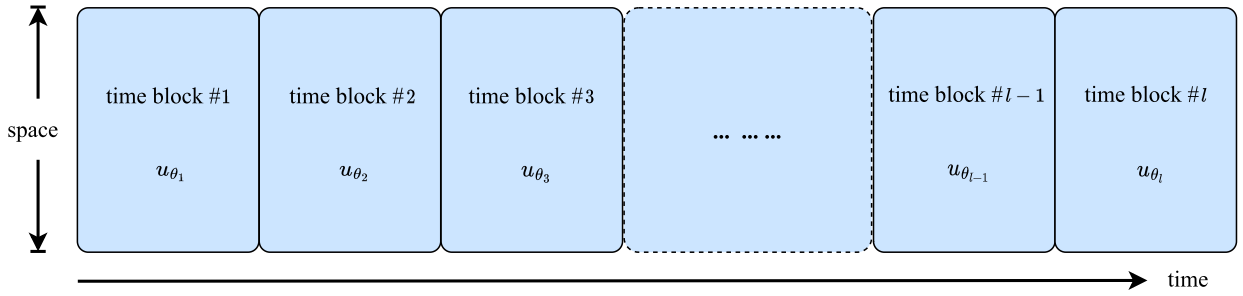


Fig. 2. Illustration of the block time marching (BTM) strategy. The large time domain is partitioned into multiple blocks, with each block computed individually and successively. Solution in one block informs the initial condition for the subsequent time block.

of all hidden nodes across the network, spanning from the first to the last hidden layers. From the logical concatenation layer to the output layer, a typical affine transformation is applied, possibly followed by an activation function, to obtain the output of the overall neural network. Notably, the logical concatenation layer does not introduce any trainable parameters. Hereafter we refer to the modified network as the hidden-layer concatenated FNN (HLConcFNN), as opposed to the original FNN, which serves as the base neural network. The incorporation of logical concatenation ensures that all the hidden nodes in the base network architecture have direct connections to the output nodes in HLConcFNN. This direct connectivity facilitates the flow of information from the hidden layers to the output layer, enhancing the network's capacity to capture intricate relations. We refer to the approach, combining physics informed neural networks with hidden-layer concatenated FNNs, as hidden-layer concatenated physics informed neural networks (HLConcPINNs).

Given architectural vector $\boldsymbol{l} = (l_0, l_1, \cdots, l_L)$, the logical concatenation layer contains a total of $N_c(\boldsymbol{l}) = \sum_{i=1}^{L-1} l_i$ virtual nodes, with the total number of hidden-layer coefficients in the neural network given by $N_h(\boldsymbol{l}) = \sum_{i=1}^{L-1}(l_{i-1}+1)l_i$. The total number of network parameters in HLConcFNN is $N_a(\boldsymbol{l}) = N_h(\boldsymbol{l}) + [N_c(\boldsymbol{l})+1]l_L$. The HLConcFNN is formally defined as the mapping $u_\theta : \mathbb{R}^{l_0} \to \mathbb{R}^{l_L}$ given by

$$u_\theta(z) = \sum_{i=1}^{L-1} M_i u_i^\vartheta(z) + b_L \qquad z \in \mathbb{R}^{l_0}, \tag{4}$$

where $\theta \in \mathbb{R}^{N_a}$ denotes all the network parameters in HLConcFNN, and $\vartheta \in \mathbb{R}^{N_h}$ denotes the hidden-layer parameters. $u_i^\vartheta(z) = \sigma \circ \mathcal{A}_i^\vartheta \circ \sigma \circ \mathcal{A}_{i-1}^\vartheta \circ \cdots \circ \sigma \circ \mathcal{A}_1^\vartheta(z)$ $(1 \le i \le L-1)$, with $M_i \in \mathbb{R}^{l_L \times l_i}$ $(1 \le i \le L-1)$ denoting the connection coefficients between the output layer and the i-th hidden layer. $b_L \in \mathbb{R}^{l_L}$ is the bias of the output layer.

Given an architectural vector $\boldsymbol{l}$ and an activation function σ, let HLConcFNN$(\boldsymbol{l}, \sigma)$ denote the hidden-layer concatenated neural network associated with this architecture. For a given domain $D \subset \mathbb{R}^d$, we define

$$U(D, \boldsymbol{l}, \sigma) = \{\ u_\theta(z) \mid u_\theta(z) \text{ is the output of HLConcFNN}(\boldsymbol{l}, \sigma),\ z \in D,\ \theta \in \mathbb{R}^{N_a(\boldsymbol{l})}\ \} \tag{5}$$

as the collection of all possible output fields of this HLConcFNN$(\boldsymbol{l}, \sigma)$. $U(D, \boldsymbol{l}, \sigma)$ denotes the set of functions that can be exactly represented by this HLConcFNN$(\boldsymbol{l}, \sigma)$ on D. Following [38], we refer to $U(D, \boldsymbol{l}, \sigma)$ as the representation capacity of the HLConcFNN$(\boldsymbol{l}, \sigma)$ for the domain D.

HLConcFNNs exhibit a hierarchical structure in terms of their representation capacity, which is crucial to the current analyses. Specifically, given a base network architecture $\boldsymbol{l}$, if a new hidden layer is appended to this network or if extra nodes are added to an existing hidden layer, the representation capacity of the HLConcFNN associated with the new architecture is guaranteed to be not smaller than that of the original architecture. These points are made precise by Lemmas 9.6 and 9.7 in Section 9.1. As a result, starting with an initial network architecture, one can attain a sequence of network architectures, by either appending one or more hidden layers to or adding extra nodes to existing hidden layers of the preceding architecture. Then we can conclude that the HLConcFNNs associated with this sequence of network architectures have non-decreasing representation capacities. By contrast, conventional FNNs do not have such a property. It is also noted that the HLConcFNN associated with a network architecture has a representation capacity at least as large as that of the conventional FNN associated with this architecture.

2.4. Block time marching (BTM)

Long-time dynamic simulation of time-dependent PDEs is a challenging issue to neural network-based methods. NN-based techniques oftentimes adopt a space-time approach for solving dynamic PDEs, in which the space and time variables are treated on the same footing. When the temporal dimension of the space-time domain becomes large, as necessitated by the interest in long-time dynamics, network training can become immensely difficult, leading to grossly inaccurate NN predictions, especially toward later time instants.

Block time marching (BTM) [14] is an effective strategy that can alleviate the challenge posed by long time horizons and facilitate accurate long-term simulations with neural networks. BTM addresses the challenge by partitioning the large time domain into multiple

windows (time blocks) and successively advancing the solution through these blocks. Fig. 2 provides a visual representation of the block time marching strategy. The space-time domain, with a long time horizon, is divided into time blocks along the time axis. Each time block should have a moderate size in time, facilitating effective capture of the dynamics. An adaptation of the time block sizes based on local error control together with related strategies (e.g. continuation technique for improved initial guesses for nonlinear solvers) can significantly enhance the BTM performance [23]. The initial-boundary value problem is then solved individually and sequentially on the space-time domain of each time block using a suitable method, in particular with HLConcPINN in this work. The solution obtained on one time block, evaluated at the last time instant, informs the initial conditions for the computation of the subsequent time block. Starting with the first time block, we march in time block by block, until the last time block is traversed.

The basic BTM formulation as described above, unfortunately, is not amenable to theoretical analysis. Our analysis requires a modification to the basic formulation, which will be discussed in subsequent sections.

2.5. Residuals and training sets

We combine block time marching and hidden-layer concatenated PINNs for solving the system (1). We provide a theoretical analysis of the resultant method, and investigate the numerical algorithms as suggested by the theory.

For simplicity we consider uniform time blocks in BTM. We divide the temporal dimension T into uniform time blocks l, where $l \geq 1$ is chosen such that the block size $\Delta T = T/l$ (Δt or ΔT, in Sections 3-6, time block $\Delta t = t_{i+1} - t_i$) is of a moderate value. Let $[t_{i-1}, t_i]$ $(1 \leq i \leq l)$ denote the i-th block in time, where $t_i = i\Delta T$ (or $t_i = i\Delta t$) and t_0 denotes the initial time. We march in time block by block, and within each time block solve the system (1) using hidden-layer concatenated PINN.

To solve (1), it is necessary to specify the residuals and the set of training collocation points. Let $\mathcal{S}_i \subset \overline{D} \times [0, t_i]$ denote the set of collocation points for training the HLConcPINN on the i-th time block $(1 \leq i \leq l)$. We define $\mathcal{S}_i = \mathcal{S}_{int_i} \cup \mathcal{S}_{sb_i} \cup \mathcal{S}_{tb_i}$ with,

- Interior training points $\mathcal{S}_{int_i} = \{\boldsymbol{y}_n, 1 \leq n \leq N_{int_i}\}$, where $\boldsymbol{y}_n = (\boldsymbol{x}_n, t_n) \in D \times (t_{i-1}, t_i)$.
- Spatial boundary training points $\mathcal{S}_{sb_i} = \{\boldsymbol{y}_n, 1 \leq n \leq N_{sb_i}\}$, where $\boldsymbol{y}_n = (\boldsymbol{x}_n, t_n) \in \partial D \times (t_{i-1}, t_i)$.
- Temporal boundary training points $\mathcal{S}_{tb_i} = \{\boldsymbol{y}_n, 1 \leq n \leq N_{tb_i}\}$, where $\boldsymbol{y}_n = (\boldsymbol{x}_n, t_n) \in D \times \{t_0, t_1, \ldots, t_{i-1}\}$.

Here, $(N_{int_i}, N_{sb_i}, N_{tb_i})$ denote the number of interior points, spatial boundary points, and temporal boundary points for the i-th block, respectively.

Define space-time domains $\Omega_i = D \times [t_{i-1}, t_i]$ and $\Omega_{*i} = \partial D \times [t_{i-1}, t_i]$ for the time block i. We employ a HLConcFNN $u_\theta : \Omega_i \to \mathbb{R}^m$ to approximate the solution u on Ω_i, and define the residual functions by $(1 \leq i \leq l)$,

$$\mathcal{R}_{int_i}[u_{\theta_i}](\boldsymbol{x}, t) = \frac{\partial u_{\theta_i}}{\partial t}(\boldsymbol{x}, t) + \mathcal{L}[u_{\theta_i}](\boldsymbol{x}, t) \qquad (\boldsymbol{x}, t) \in \Omega_i, \tag{6a}$$

$$\mathcal{R}_{sb_i}[u_{\theta_i}](\boldsymbol{x}, t) = \mathcal{B}u_{\theta_i}(\boldsymbol{x}, t) - u_d(\boldsymbol{x}, t) \qquad (\boldsymbol{x}, t) \in \Omega_{*i}, \tag{6b}$$

$$\mathcal{R}_{tb_i}[u_{\theta_i}](\boldsymbol{x}, t_{i-1}) = u_{\theta_i}(\boldsymbol{x}, t_{i-1}) - u_{\theta_{i-1}}(\boldsymbol{x}, t_{i-1}) \qquad \boldsymbol{x} \in D, \tag{6c}$$

where $u_{\theta_0}(\boldsymbol{x}, t_0) = u_{in}(\boldsymbol{x})$. These residuals characterize the extent to which a given function u satisfies the initial/boundary value problem (1) for the time block i. If u is the exact solution, then $\mathcal{R}_{int_i}[u] = \mathcal{R}_{sb_i}[u] = \mathcal{R}_{tb_1}[u] \equiv 0$. The above settings on the time block partitions and training sets will be employed throughout the subsequent sections.

In the forthcoming sections, we focus on four time-dependent partial differential equations: the heat equation, the viscous Burgers' equation, the wave equation and the nonlinear Klein-Gordon equation, representative of parabolic and hyperbolic type PDEs. We provide an analysis of the HLConcPINN method for approximating the solutions to these equations for long-time dynamic simulations, and investigate the PINN type computational algorithm stemming from these analyses. We implement these algorithms and numerically demonstrate the effectiveness of the HLConcPINN method using extensive experiments. The analyses herein of the HLConcPINN method, excluding its BTM component, equally apply to stationary PDEs of the elliptical type. An analysis for the nonlinear Helmholtz equation is provided in Appendix 9.3.

3. HLConcPINN for approximating the heat equation

3.1. Heat equation

Let $D \subset \mathbb{R}^d$ denote an open connected bounded domain with a C^k boundary ∂D. We consider the heat equation:

$$\frac{\partial u(\boldsymbol{x}, t)}{\partial t} - \Delta u(\boldsymbol{x}, t) = f(\boldsymbol{x}, t) \qquad (\boldsymbol{x}, t) \in D \times [0, T], \tag{7a}$$

$$u(\boldsymbol{x}, 0) = u_{in}(\boldsymbol{x}) \qquad \boldsymbol{x} \in D, \tag{7b}$$

$$u(\boldsymbol{x}, t) = u_d(\boldsymbol{x}, t) \qquad (\boldsymbol{x}, t) \in \partial D \times [0, T]. \tag{7c}$$

Here, u is the field solution, f is a source term, and u_{in} and u_d denote the initial distribution and the boundary data, respectively.

Remark 3.1. The existence and regularity of the solution to the heat equation can be found in [48,21]. Assuming homogeneous Dirichlet boundary condition (i.e. $u_d = 0$), the following results hold:

- [48, Theorem 3.1, Chapter II]: For u_{in} given in $L^2(D)$ and f given in $L^2(0,T;H^{-1}(D))$, there exists a unique solution u of (7) such that

$$u \in L^2(0,T;H_0^1(D)) \cap C(0,T;L^2(D)), \quad \frac{\partial u}{\partial t} \in L^2(0,T;H^{-1}(D)).$$

- [48, Theorem 3.3, Chapter II] and [21, Theorem 5, Chapter 7.1]: Assume that $u_{in} \in H_0^1(D)$ and $f \in L^2(0,T;L^2(D))$, then

$$u \in L^2(0,T;H^2(D) \cap H_0^1(D)) \cap C(0,T;H_0^1(D)), \quad \frac{\partial u}{\partial t} \in L^2(0,T;L^2(D)).$$

- [21, Theorem 6, Chapter 7.1]: The regularity result for higher-order regular initial data and source terms is available.

3.2. Hidden-layer concatenated physics informed neural networks

We divide the temporal domain into l blocks, and seek l deep neural networks $u_{\theta_i} : D \times [0,t_i] \to \mathbb{R}$, parameterized by θ_i, to approximate the solution u of (7) for $1 \le i \le l$. For any $u_{\theta_i} : D \times [0,t_i] \to \mathbb{R}$ $(1 \le i \le l)$, we define the residuals:

$$R_{int_i}[u_{\theta_i}](\boldsymbol{x},t) = \frac{\partial u_{\theta_i}}{\partial t} - \Delta u_{\theta_i} - f, \tag{8a}$$

$$R_{tb_i}[u_{\theta_i}](\boldsymbol{x},t_{j-1}) = u_{\theta_i}(\boldsymbol{x},t_{j-1}) - u_{\theta_{j-1}}(\boldsymbol{x},t_{j-1}) \qquad j = 1,2,\cdots,i, \tag{8b}$$

$$R_{sb_i}[u_{\theta_i}](\boldsymbol{x},t) = u_{\theta_i}(\boldsymbol{x},t) - u_d(\boldsymbol{x},t), \tag{8c}$$

Here $u_{\theta_0}(\boldsymbol{x},t_0) = u_{in}(\boldsymbol{x})$, and Ω_i and Ω_{*i} are defined in Section 2.5. Note that for the exact solution $R_{int_i}[u] = R_{tb_i}[u] = R_{sb_i}[u] = 0$.

With HLConcPINN, we try to find a sequence of neural networks u_{θ_i} $(1 \le i \le l)$, for which all the residuals are minimized. Specifically, we minimize the quantity,

$$\mathcal{E}_{G_i}(\theta_i)^2 = \widetilde{\mathcal{E}}_{G_i}(\theta_i)^2 + \mathcal{E}_{G_{i-1}}(\theta_{i-1})^2, \tag{9}$$

sequentially for $1 \le i \le l$, where

$$\widetilde{\mathcal{E}}_{G_i}(\theta_i)^2 = \int_{\Omega_i} |R_{int_i}[u_{\theta_i}](\boldsymbol{x},t)|^2 \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t + \sum_{j=1}^{i} \int_D |R_{tb_i}[u_{\theta_i}](\boldsymbol{x},t_{j-1})|^2 \,\mathrm{d}\boldsymbol{x} + \left(\int_{\Omega_{*i}} |R_{sb_i}[u_{\theta_i}](\boldsymbol{x},t)|^2 \,\mathrm{d}s(\boldsymbol{x})\,\mathrm{d}t \right)^{\frac{1}{2}}. \tag{10}$$

In equation (9) we set $\mathcal{E}_{G_{i-1}}(\theta_{i-1}) = 0$ for $i = 1$. The quantity $\mathcal{E}_{G_i}(\theta)$ is commonly known as the population risk or generalization error of the neural networks u_{θ_i}.

Remark 3.2. In the original block time marching scheme from [14], when computing a particular time block, the initial condition is taken to be the solution data from the preceding time block evaluated at the last time instant. In light of [48,21] (see Remark 3.1), the regularity of the initial data influences the regularity of the solution. By Lemma 9.8 (in the appendix), the regularity of this initial value $\hat{u}_{\theta_i}|_{t=t_{i-1}}$ differs from the original initial data of the problem. This difference affects the regularity of the PDE solution in the current time block, as well as the approximation accuracy of the neural network solution, causing difficulty in the analysis.

To address this issue, we make the following crucial modification to block time marching. We employ the true initial data for the problem as the initial value for all time blocks within the interval $[0,t_i]$ for $1 \le i \le l$, as specified by (8b). This ensures that the regularity of the initial value is maintained throughout the time blocks, preserving the regularity of the solution. Essentially, we enforce the PDE and the boundary conditions only on the interval $[t_{i-1},t_i]$ in time. For the time periods $[0,t_{i-1}]$, however, we enforce the residuals solely at the discrete points t_{j-1} $(1 \le j \le i)$. By using the true initial data consistently and training the neural network within individual time blocks successively, we can maintain the regularity of the solution across all time blocks. The initial condition (8b) and the setting for training data points in subsequent discussions employ this modified BTM formulation.

The integrals in (9) can be approximated numerically, leading to a training loss function. Following the discussions of Section 2.5, the full training set consists of $\mathcal{S} = \bigcup_{i=1}^{l} \mathcal{S}_i$ with $\mathcal{S}_i = \mathcal{S}_{int_i} \cup \mathcal{S}_{sb_i} \cup \mathcal{S}_{tb_i}$ and we employ the midpoint rule for the numerical quadrature. This leads to the following approximation:

$$\mathcal{E}_{T_i}(\theta_i,\mathcal{S}_i)^2 = \widetilde{\mathcal{E}}_{T_i}(\theta_i,\mathcal{S}_i)^2 + \mathcal{E}_{T_{i-1}}(\theta_{i-1},\mathcal{S}_{i-1})^2, \tag{11}$$

$$\widetilde{\mathcal{E}}_{T_i}(\theta_i,\mathcal{S}_i)^2 = \mathcal{E}_T^{int_i}(\theta_i,\mathcal{S}_{int_i})^2 + \mathcal{E}_T^{tb_i}(\theta_i,\mathcal{S}_{tb_i})^2 + \mathcal{E}_T^{sb_i}(\theta_i,\mathcal{S}_{sb_i}), \tag{12}$$

where

$$\mathcal{E}_T^{int_i}(\theta_i, S_{int_i})^2 = \sum_{n=1}^{N_{int_i}} \omega_{int_i}^n |R_{int_i}[u_{\theta_i}](\boldsymbol{x}_{int_i}^n, t_{int_i}^n))|^2, \tag{13a}$$

$$\mathcal{E}_T^{sb_i}(\theta_i, S_{sb_i})^2 = \sum_{n=1}^{N_{sb_i}} \omega_{sb_i}^n |R_{sb_i}[u_{\theta_i}](\boldsymbol{x}_{sb_i}^n, t_{sb_i}^n))|^2, \tag{13b}$$

$$\mathcal{E}_T^{tb_i}(\theta_i, S_{tb_i})^2 = \sum_{j=1}^{i} \sum_{n=1}^{N_{tb_i}} \omega_{tb_i}^n |R_{tb_i}[u_{\theta_i}](\boldsymbol{x}_{tb_i}^n, t_{j-1})|^2, \tag{13c}$$

with the term $\mathcal{E}_{T_{i-1}}(\theta_{i-1}, S_{i-1}) = 0$ for $i = 1$. Here, the quadrature points in space-time constitute the data sets $S_{int_i} = \{(\boldsymbol{x}_{int_i}^n, t_{int_i}^n)\}_{n=1}^{N_{int_i}}$, $S_{tb_i} = \{\boldsymbol{x}_{tb_i}^n\}_{n=1}^{N_{tb_i}}$ and $S_{sb_i} = \{(\boldsymbol{x}_{sb_i}^n, t_{sb_i}^n)\}_{n=1}^{N_{sb_i}}$, and $\omega_{\star_i}^n$ are the quadrature weights with $\star$ denoting *int*, *tb* or *sb*.

3.3. Error analysis

Let $\hat{u}_i = u_{\theta_i} - u$ denote the error of the HLConcPINN approximation (u_{θ_i}) against the true solution (u). By using equation (7) and definitions of the residuals (8), we obtain

$$R_{int_i} = \frac{\partial \hat{u}_i}{\partial t} - \Delta \hat{u}_i, \tag{14a}$$

$$R_{tb_i}|_{t=t_{j-1}} = \hat{u}_i|_{t=t_{j-1}} - \hat{u}_{j-1}|_{t=t_{j-1}} \qquad j = 1, 2, \cdots, i, \tag{14b}$$

$$R_{sb_i} = \hat{u}_i|_{\partial D}, \tag{14c}$$

where $\hat{u}_0|_{t=t_0} = 0$. We define the total error of the HLConcPINN approximation by

$$\mathcal{E}(\theta_i)^2 = \int_{t_{i-1}}^{t_i} \int_D |\hat{u}_i(\boldsymbol{x}, t)|^2 \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}t \quad 1 \le i \le l. \tag{15}$$

The bounds on the HLConcPINN residuals and its approximation errors are provided by the following three theorems. The proofs for these theorems are given in the appendix (Section 9.2).

Theorem 3.3. *Let $\widetilde{\Omega}_i = D \times [0, t_i]$ and $\widetilde{\Omega}_{*i} = \partial D \times [0, t_i]$. Suppose n, d, $k \in \mathbb{N}$ with $n \ge 2$ and $k \ge 3$, and $u \in H^k(\widetilde{\Omega}_i)$. For every integer $N > 5$, there exists a HLConcPINN u_{θ_i} such that*

$$\|R_{int_i}\|_{L^2(\widetilde{\Omega}_i)} \lesssim N^{-k+2} \ln^2 N; \qquad \|R_{tb_i}(\boldsymbol{x}, t_{j-1})\|_{L^2(D)}, \ \|R_{sb_i}\|_{L^2(\widetilde{\Omega}_{*i})} \lesssim N^{-k+1} \ln N \quad 1 \le j \le i. \tag{16}$$

Theorem 3.3 implies that one can make the HLConcPINN residuals (8) arbitrarily small by choosing N to be sufficiently large. It follows that the generalization error $\mathcal{E}_{G_i}(\theta_i)^2$ in (9) can be made arbitrarily small.

The next two theorems indicate that the approximation error $\mathcal{E}(\theta_i)^2$ is also small when the generalization error $\mathcal{E}_{G_i}(\theta_i)^2$ is small with the HLConcPINN approximation u_{θ_i}. Moreover, the approximation error $\mathcal{E}(\theta_i)^2$ can be arbitrarily small, provided that the training error $\mathcal{E}_{T_i}(\theta_i, S_i)^2$ is sufficiently small and the sample set is sufficiently large.

Theorem 3.4. *Let $d \in \mathbb{N}$, and $u \in C^1(\widetilde{\Omega}_i)$ be the classical solution to (7). Let u_{θ_i} be a HLConcPINN with parameter θ_i, $t_{i-1} \le \tau \le t_i$, and $\Delta t = t_i - t_{i-1}$ (time block size). Then the following relation holds,*

$$\int_D |\hat{u}_i(\boldsymbol{x}, \tau)|^2 \, \mathrm{d}\boldsymbol{x} \le C_{G_i} \exp(\Delta t), \qquad \int_{t_{i-1}}^{t_i} \int_D |\hat{u}_i(\boldsymbol{x}, t)|^2 \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}t \le C_{G_i} \Delta t \exp(\Delta t), \tag{17}$$

where

$$C_{G_i} = \widetilde{C}_{G_i} + 2C_{G_{i-1}} \exp(\Delta t), \qquad C_{G_0} = 0,$$

$$\widetilde{C}_{G_i} = 2 \sum_{j=1}^{i} \int_D |R_{tb_i}(\boldsymbol{x}, t_{j-1})|^2 \, \mathrm{d}\boldsymbol{x} + \int_{t_{i-1}}^{t_i} \int_D |R_{int_i}|^2 \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}t + 2C_{\partial D_i} |\Delta t|^{\frac{1}{2}} \Big(\int_{t_{i-1}}^{t_i} \int_{\partial D} |R_{sb_i}|^2 \, \mathrm{d}s(\boldsymbol{x}) \mathrm{d}t \Big)^{\frac{1}{2}},$$

and $C_{\partial D_i} = |\partial D|^{\frac{1}{2}} \big(\|u\|_{C^1(\Omega_{*i})} + \|u_{\theta_i}\|_{C^1(\Omega_{*i})} \big)$. The symbol $\Omega_{*i} = \partial D \times [t_{i-1}, t_i]$ is given in Section 2.5.

As described in Section 2.2, we focus on training sets that adopt the midpoint rule $\mathcal{Q}_M^{\Lambda}[f]$ (see (3)) in our analysis. For $f = R_{int_i}^2$ and $\Lambda = \Omega_i = D \times [t_{i-1}, t_i]$, we have the quadrature $\mathcal{Q}_{M_{int_i}}^{\Omega_i}[R_{int_i}^2]$. Similarly, we can obtain the quadrature $\mathcal{Q}_{M_{sb_i}}^{\Omega_{*i}}[R_{sb_i}^2]$ for $f = R_{sb_i}^2$ and $\Lambda = \Omega_{*i} = \partial D \times [t_{i-1}, t_i]$, and the quadrature $\mathcal{Q}_{M_{tb_i}}^{D}[R_{tb_i}^2]$ for $f = R_{tb_i}^2$ and $\Lambda = D$. We use M_{int_i}, M_{sb_i} and M_{tb_i} to denote the number of quadrature points for Ω_i, Ω_{*i} and D, respectively. Then, the loss function (12) can be rewritten as

$$\begin{aligned}\widetilde{\mathcal{E}}_{T_i}(\theta_i, S_i)^2 &= \mathcal{E}_T^{int_i}(\theta_i, S_{int_i})^2 + \mathcal{E}_T^{tb_i}(\theta_i, S_{tb_i})^2 + \mathcal{E}_T^{sb_i}(\theta_i, S_{sb_i}) \\ &= \mathcal{Q}_{M_{int_i}}^{\Omega_i}[R_{int_i}^2] + \sum_{j=1}^{i} \mathcal{Q}_{M_{tb_i}}^{D}[R_{tb_i}^2(\boldsymbol{x}, t_{j-1})] + (\mathcal{Q}_{M_{sb_i}}^{\Omega_{*i}}[R_{sb_i}^2])^{\frac{1}{2}}.\end{aligned} \tag{18}$$

Using the above notation, relation (3), and Theorem 3.4, we can prove below that the approximation error can be made arbitrarily small, with a sufficiently small training error and a sufficiently large sample set.

Theorem 3.5. *Let $d \in \mathbb{N}$ and $T > 0$. Let $u \in C^4(\widetilde{\Omega}_i)$ be the classical solution to (7), and u_{θ_i} $(1 \le i \le l)$ be a HLConcPINN with parameter θ_i. Then the total error satisfies*

$$\begin{aligned}&\int_{t_{i-1}}^{t_i} \int_D |\hat{u}_i(\boldsymbol{x}, t)|^2 \, \mathrm{d}\boldsymbol{x}\, \mathrm{d}t \le C_{T_i} \Delta t \exp(\Delta t) \\ &= \mathcal{O}\left(\mathcal{E}_{T_i}(\theta_i, S_i)^2 + M_{int_i}^{-\frac{2}{d+1}} + M_{tb_i}^{-\frac{2}{d}} + M_{sb_i}^{-\frac{1}{d}} \right),\end{aligned} \tag{19}$$

where $\mathcal{O}$ denotes the same order of magnitude and the constant C_{T_i} is defined by

$$C_{T_i} = \widetilde{C}_{T_i} + 2C_{T_{i-1}} \exp(\Delta t), \qquad C_{T_0} = 0, \tag{20}$$

$$\begin{aligned}\widetilde{C}_{T_i} &= 2 \sum_{j=1}^{i} \left(C_{(R_{tb_i}^2(\boldsymbol{x}, t_{j-1}))} M_{tb_i}^{-\frac{2}{d}} + \mathcal{Q}_{M_{tb_i}}^{D}[R_{tb_i}^2(\boldsymbol{x}, t_{j-1})] \right) \\ &\quad + C_{(R_{int_i}^2)} M_{int_i}^{-\frac{2}{d+1}} + \mathcal{Q}_{M_{int_i}}^{\Omega_i}[R_{int_i}^2] + 2C_{\partial D_i} |\Delta t|^{\frac{1}{2}} \left(C_{(R_{sb_i}^2)} M_{sb_i}^{-\frac{2}{d}} + \mathcal{Q}_{M_{sb_i}}^{\Omega_{*i}}[R_{sb_i}^2] \right)^{\frac{1}{2}}.\end{aligned}$$

Here $C_{(f^2)}$ is a constant related to f^2 and determined by its boundedness, with f being R_{tb_i}, R_{int_i}, or R_{sb_i}.

4. HLConcPINN for approximating the Burgers' equation

4.1. Viscous Burgers' equation

We consider the 1D viscous Burgers' equation on the domain $D = [a, b] \subset \mathbb{R}$:

$$\frac{\partial u(x,t)}{\partial t} - \nu \frac{\partial^2 u(x,t)}{\partial x^2} + u(x,t) \frac{\partial u(x,t)}{\partial x} = f(x,t) \qquad (x,t) \in D \times [0,T], \tag{21a}$$

$$u(x,0) = u_{in}(x) \qquad x \in D, \tag{21b}$$

$$u(a,t) = g_1(t), \qquad u(b,t) = g_2(t), \qquad t \in [0,T], \tag{21c}$$

where the constant ν denotes the viscosity, f is a prescribed source term, $g_1(t)$ and $g_2(t)$ denote the boundary data, and $u_{in}(x)$ is the initial distribution.

4.2. Hidden-layer concatenated physics informed neural networks

We follow the settings from Section 2.5, and seek deep neural networks $u_{\theta_i} : D \times [0, t_i] \to \mathbb{R}$ for $1 \le i \le l$ (l denoting the number of time blocks) to approximate the solution u of (21). Define the following residual functions, for $1 \le i \le l$,

$$R_{int_i}[u_{\theta_i}](x,t) = \frac{\partial u_{\theta_i}}{\partial t} - \nu \frac{\partial^2 u_{\theta_i}}{\partial x^2} + u_{\theta_i} \frac{\partial u_{\theta_i}}{\partial x} - f, \tag{22a}$$

$$R_{tb_i}[u_{\theta_i}](x, t_{j-1}) = u_{\theta_i}|_{t=t_{j-1}} - u_{\theta_{j-1}}|_{t=t_{j-1}} \qquad 1 \le j \le i, \tag{22b}$$

$$R_{sb1_i}[u_{\theta_i}](a,t) = u_{\theta_i}(a,t) - g_1(t), \qquad R_{sb2_i}[u_{\theta_i}](b,t) = u_{\theta_i}(b,t) - g_2(t). \tag{22c}$$

In these equations $u_{\theta_0}\big|_{t=t_0} = u_{in}(x)$. Note that $R_{int_i}[u] = R_{tb_i}[u] = R_{sb_i}[u] = 0$ for the exact solution u. With HLConcPINN we seek θ_i $(1 \le i \le l)$ to minimize the following quantity,

$$\mathcal{E}_{G_i}(\theta_i)^2 = \widetilde{\mathcal{E}}_{G_i}(\theta_i)^2 + \mathcal{E}_{G_{i-1}}(\theta_{i-1})^2 \quad 1 \le i \le l, \tag{23}$$

$$\begin{aligned}\widetilde{\mathcal{E}}_{G_i}(\theta_i)^2 = &\int_{t_{i-1}}^{t_i}\int_D |R_{int_i}[u_{\theta_i}](x,t)|^2\,\mathrm{d}x\,\mathrm{d}t + \int_{t_{i-1}}^{t_i}(|R_{sb1_i}[u_{\theta_i}](a,t)|^2 + |R_{sb2_i}[u_{\theta_i}](b,t)|^2)\,\mathrm{d}t\\ &+\left(\int_{i-1}^{t_i}|R_{sb1_i}[u_{\theta_i}](a,t)|^2\,\mathrm{d}t\right)^{\frac{1}{2}} + \left(\int_{i-1}^{t_i}|R_{sb2_i}[u_{\theta_i}](b,t)|^2\,\mathrm{d}t\right)^{\frac{1}{2}}\\ &+\sum_{j=1}^{i}\int_D |R_{tb_i}[u_{\theta_i}](x,t_{j-1})|^2\,\mathrm{d}x,\end{aligned} \tag{24}$$

where $\mathcal{E}_{G_{i-1}}(\theta_{i-1}) = 0$ for $i = 1$.

The training data set consists of $\mathcal{S} = \bigcup_{i=1}^{l}\mathcal{S}_i$, with $\mathcal{S}_i = \mathcal{S}_{int_i} \cup \mathcal{S}_{sb_i} \cup \mathcal{S}_{tb_i}$. The spatial boundary training points are $\mathcal{S}_{sb_i} = \{y_n\}$ for $1 \le n \le N_{sb_i}$, with $y_n = (x,t)_n \in \{a,b\} \times (t_{i-1}, t_i)$. We approximate the integrals in (23) by the mid-point rule, leading to the training loss functions,

$$\mathcal{E}_{T_i}(\theta_i, \mathcal{S}_i)^2 = \widetilde{\mathcal{E}}_{T_i}(\theta_i, \mathcal{S}_i)^2 + \mathcal{E}_{T_{i-1}}(\theta_{i-1}, \mathcal{S}_{i-1})^2 \quad 1 \le i \le l, \tag{25}$$

$$\begin{aligned}\widetilde{\mathcal{E}}_{T_i}(\theta_i, \mathcal{S}_i)^2 = &\,\mathcal{E}_T^{int_i}(\theta_i, \mathcal{S}_{int_i})^2 + \mathcal{E}_T^{sb1_i}(\theta_i, \mathcal{S}_{sb_i})^2 + \mathcal{E}_T^{sb2_i}(\theta_i, \mathcal{S}_{sb_i})^2\\ &+ \mathcal{E}_T^{sb1_i}(\theta_i, \mathcal{S}_{sb_i}) + \mathcal{E}_T^{sb2_i}(\theta_i, \mathcal{S}_{sb_i}) + \mathcal{E}_T^{tb_i}(\theta_i, \mathcal{S}_{tb_i})^2,\end{aligned} \tag{26}$$

where $\mathcal{E}_T^{sb1_i}(\theta_i, \mathcal{S}_{sb_i})^2 = \sum_{n=1}^{N_{sb_i}} \omega_{sb_i}^n |R_{sb_i}[u_{\theta_i}](a, t_{sb_i}^n)|^2$, $\mathcal{E}_T^{sb2_i}(\theta_i, \mathcal{S}_{sb_i})^2 = \sum_{n=1}^{N_{sb_i}} \omega_{sb_i}^n |R_{sb_i}[u_{\theta_i}](b, t_{sb_i}^n)|^2$, and the remaining terms are defined according to equation (13). Note that $\mathcal{E}_{T_{i-1}}(\theta_{i-1}, \mathcal{S}_{i-1}) = 0$ for $i = 1$.

4.3. Error analysis

Let $\hat{u}_i = u_{\theta_i} - u$ denote the error of the HLConcPINN approximation (u denoting the exact solution). Applying the Burgers' equation (21) and the definitions of the different residuals, we obtain for $1 \le i \le l$,

$$R_{int_i} = \frac{\partial \hat{u}_i}{\partial t} - \nu\frac{\partial^2 \hat{u}_i}{\partial x^2} + u_{\theta_i}\frac{\partial u_{\theta_i}}{\partial x} - u\frac{\partial u}{\partial x}, \tag{27a}$$

$$R_{tb_i}|_{t=t_{j-1}} = \hat{u}_i|_{t=t_{j-1}} - \hat{u}_{j-1}|_{t=t_{j-1}} \qquad j = 1, 2, \cdots, i, \tag{27b}$$

$$R_{sb1_i}(a,t) = \hat{u}_i(a,t), \qquad R_{sb2_i}(b,t) = \hat{u}_i(b,t), \tag{27c}$$

where $\hat{u}_0|_{t=t_0} = 0$. Then, we define the total error of the HLConcPINN approximation as

$$\mathcal{E}(\theta_i)^2 = \int_{t_{i-1}}^{t_i}\int_D |\hat{u}_i(\boldsymbol{x}, t)|^2\,\mathrm{d}x\,\mathrm{d}t. \tag{28}$$

Theorem 4.1. *Let $\widetilde{\Omega}_i = D \times [0, t_i]$. Suppose n, d, $k \in \mathbb{N}$ with $n \ge 2$ and $k \ge 3$, and $u \in H^k(\widetilde{\Omega}_i)$. For every integer $N > 5$, there exists a HLConcPINN u_{θ_i} such that*

$$\|R_{int_i}\|_{L^2(\widetilde{\Omega}_i)} \lesssim N^{-k+2}\ln^2 N, \tag{29a}$$

$$\|R_{tb_i}(x, t_{j-1})\|_{L^2(D)}, \|R_{sb1_i}\|_{L^2(\{a\}\times[0,t_i])}, \|R_{sb2_i}\|_{L^2(\{b\}\times[0,t_i])} \lesssim N^{-k+1}\ln N, \quad 1 \le j \le i. \tag{29b}$$

Proof. By applying $u \in H^k(\widetilde{\Omega}_i)$, Lemmas 9.2 and 9.8, we can conclude the proof. □

Theorem 4.2. *Let $u \in C^1(\widetilde{\Omega}_i)$ be the classical solution to (21). Let u_{θ_i} $(1 \le i \le l)$ be a HLConcPINN with parameter θ_i. Then the following relation holds,*

$$\int_D |\hat{u}_i(x,\tau)|^2\,\mathrm{d}x \le C_{G_i}\exp((1 + C_{D_i})\Delta t) \quad \tau \in [t_{i-1}, t_i], \tag{30}$$

$$\int_{t_{i-1}}^{t_i}\int_D |\hat{u}_i(x,t)|^2\,\mathrm{d}x\,\mathrm{d}t \le C_{G_i}\Delta t\exp((1 + C_{D_i})\Delta t), \tag{31}$$

where

$$C_{G_i} = 2C_{G_{i-1}} \exp((1 + C_{D_{i-1}})\Delta t) + \widetilde{C}_{G_i}, \qquad C_{G_0} = 0, \qquad C_{D_0} = 0,$$

$$\widetilde{C}_{G_i} = 2\sum_{j=1}^{i} \int_D |R_{tb_i}(x, t_{j-1})|^2 \,\mathrm{d}x + \int_{t_{i-1}}^{t_i} \int_D |R_{int_i}|^2 \,\mathrm{d}x\,\mathrm{d}t + C_{\partial D_{1i}} \Big(\int_{t_{i-1}}^{t_i} |R_{sb1_i}|^2 \,\mathrm{d}t \Big)^{\frac{1}{2}}$$
$$+ C_{\partial D_{1i}} \Big(\int_{t_{i-1}}^{t_i} |R_{sb2_i}|^2 \,\mathrm{d}t \Big)^{\frac{1}{2}} + C_{\partial D_{2i}} \int_{t_{i-1}}^{t_i} \big(|R_{sb1_i}|^2 + |R_{sb2_i}|^2 \big) \,\mathrm{d}t,$$

$C_{\partial D_{1i}} = 2\nu\Delta t^{\frac{1}{2}} (\|u\|_{C^1(\Omega_{*i})} + \|u_{\theta_i}\|_{C^1(\Omega_{*i})})$, $C_{D_i} = 2\Delta t^{\frac{1}{2}} \big(\|u_{\theta_i}\|_{C^1(\Omega_i)} + \frac{1}{2}\|u\|_{C^1(\Omega_i)} \big)$, *and* $C_{\partial D_{2i}} = \Delta t^{\frac{1}{2}} \|u\|_{C^0(\Omega_{*i})}$ *with* $\Omega_i = D \times [t_{i-1}, t_i]$ *and* $\Omega_{*i} = \partial D \times [t_{i-1}, t_i]$.

Proof. Equation (27a) can be re-written as

$$R_{int_i} = \frac{\partial \hat{u}_i}{\partial t} - \nu \frac{\partial^2 \hat{u}_i}{\partial x^2} + \hat{u}_i \frac{\partial \hat{u}_i}{\partial x} + \hat{u}_i \frac{\partial u}{\partial x} + u \frac{\partial \hat{u}_i}{\partial x}. \tag{32}$$

Note the following relation,

$$\int_D u \frac{\partial \hat{u}_i}{\partial x} \hat{u}_i \,\mathrm{d}x = \frac{1}{2} \int_D u \frac{\partial \hat{u}_i^2}{\partial x} \,\mathrm{d}x = \frac{1}{2} u \hat{u}_i^2 \Big|_a^b - \frac{1}{2} \int_D |\hat{u}_i^2| \frac{\partial u}{\partial x} \,\mathrm{d}x.$$

The rest of the proof follows the same approach in the proof of Theorem 3.4. □

Theorem 4.3. *Let* $u \in C^4(\widetilde{\Omega}_i)$ *be the classical solution of the Burgers' equation* (21)*, and let* u_{θ_i} $(1 \le i \le l)$ *be a HLConcPINN with parameter* θ_i*. Then the total approximation error satisfies*

$$\int_{t_{i-1}}^{t_i} \int_D |\hat{u}_i(x,t)|^2 \,\mathrm{d}x\,\mathrm{d}t \le C_{T_i} \Delta t \exp((1 + C_{D_i})\Delta t)$$
$$= \mathcal{O}(\mathcal{E}_{T_i}(\theta_i, S_i)^2 + M_{int_i}^{-\frac{2}{d+1}} + M_{tb_i}^{-\frac{2}{d}} + M_{sb_i}^{-\frac{1}{d}}), \tag{33}$$

where C_{D_i} *is defined in Theorem 4.2 and*

$$C_{T_i} = \widetilde{C}_{T_i} + 2C_{T_{i-1}} \exp((1 + C_{D_{i-1}})\Delta t), \qquad C_{T_0} = 0, \tag{34}$$

$$\widetilde{C}_{T_i} = 2\sum_{j=1}^{i} \Big(C_{(R^2_{tb_i}(x,t_{j-1}))} M_{tb_i}^{-\frac{2}{d}} + \mathcal{Q}^D_{M_{tb_i}}[R^2_{tb_i}(x,t_{j-1})] \Big) + C_{(R^2_{int_i})} M_{int_i}^{-\frac{2}{d+1}} + \mathcal{Q}^{\Omega_i}_{M_{int_i}}[R^2_{int_i}]$$
$$+ C_{\partial D_{1i}} \Big(C_{(R^2_{sb1_i})} M_{sb_i}^{-\frac{2}{d}} + \mathcal{Q}^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb1_i}] \Big)^{\frac{1}{2}} + C_{\partial D_{1i}} \Big(C_{(R^2_{sb2_i})} M_{sb_i}^{-\frac{2}{d}} + \mathcal{Q}^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb2_i}] \Big)^{\frac{1}{2}}$$
$$+ C_{\partial D_{2i}} \Big(C_{(R^2_{sb1_i})} M_{sb_i}^{-\frac{2}{d}} + \mathcal{Q}^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb1_i}] + C_{(R^2_{sb2_i})} M_{sb_i}^{-\frac{2}{d}} + \mathcal{Q}^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb2_i}] \Big). \tag{35}$$

Similar to (18)*, here the quadrature* $\mathcal{Q}^{\Lambda}_M[f]$ *is defined for different functions* ($f = R^2_{tb_i}$, $R^2_{int_i}$, $R^2_{sb1_i}$ *and* R_{sb2_i})*, different domains* ($\Lambda = \Omega_i$, Ω_{*i} *and* D) *and the corresponding numbers of quadrature points* ($M = M_{int_i}$, M_{sb_i} *and* M_{tb_i}).

Proof. The proof follows from Lemma 9.3, Theorem 4.2, and the quadrature error formula (3). □

5. HLConcPINN for approximating the wave equation

5.1. Wave equation

Consider the wave equation on the torus $D = [0, 1)^d \subset \mathbb{R}^d$ with periodic boundary conditions:

$$\frac{\partial u(\boldsymbol{x}, t)}{\partial t} - v(\boldsymbol{x}, t) = 0 \qquad (\boldsymbol{x}, t) \in D \times [0, T], \tag{36a}$$

$$\frac{\partial v(\boldsymbol{x}, t)}{\partial t} - \Delta u(\boldsymbol{x}, t) = f(\boldsymbol{x}, t) \qquad (\boldsymbol{x}, t) \in D \times [0, T], \tag{36b}$$

$$u(\boldsymbol{x}, 0) = \psi_1(\boldsymbol{x}) \qquad \boldsymbol{x} \in D, \tag{36c}$$

$$v(\boldsymbol{x}, 0) = \psi_2(\boldsymbol{x}) \qquad \boldsymbol{x} \in D, \tag{36d}$$

$$u(\boldsymbol{x},t) = u(\boldsymbol{x}+1,t) \qquad (\boldsymbol{x},t) \in \partial D \times [0,T], \tag{36e}$$

$$\nabla u(\boldsymbol{x},t) = \nabla u(\boldsymbol{x}+1,t) \qquad (\boldsymbol{x},t) \in \partial D \times [0,T], \tag{36f}$$

where (u,v) are the field functions to solve, f is a source term, and (ψ_1,ψ_2) denote the initial data for (u,v).

5.2. Hidden-layer concatenated physics informed neural networks

Following the settings from Section 2.5, we seek neural networks $u_{\theta_i} : D \times [0,t_i] \to \mathbb{R}$ and $v_{\theta_i} : D \times [0,t_i] \to \mathbb{R}$ with $1 \le i \le l$, parameterized by θ_i, that approximate the solutions u and v of (36). We define the residuals (for $1 \le i \le l$ and $1 \le j \le i$),

$$R_{int1_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t) = \frac{\partial u_{\theta_i}}{\partial t} - v_{\theta_i}, \quad R_{int2_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t) = \frac{\partial v_{\theta_i}}{\partial t} - \Delta u_{\theta_i} - f, \tag{37a}$$

$$R_{tb1_i}[u_{\theta_i}](\boldsymbol{x},t_{j-1}) = u_{\theta_i}(\boldsymbol{x},t_{j-1}) - u_{\theta_{j-1}}(\boldsymbol{x},t_{j-1}), \quad R_{tb2_i}[v_{\theta_i}](\boldsymbol{x},t_{j-1}) = v_{\theta_i}(\boldsymbol{x},t_{j-1}) - v_{\theta_{j-1}}(\boldsymbol{x},t_{j-1}), \tag{37b}$$

$$R_{sb1_i}[v_{\theta_i}](\boldsymbol{x},t) = v_{\theta_i}(\boldsymbol{x},t) - v_{\theta_i}(\boldsymbol{x}+1,t), \quad R_{sb2_i}[u_{\theta_i}](\boldsymbol{x},t) = \nabla u_{\theta_i}(\boldsymbol{x},t) - \nabla u_{\theta_i}(\boldsymbol{x}+1,t), \tag{37c}$$

where $u_{\theta_0}(\boldsymbol{x},t_0) = \psi_1(\boldsymbol{x})$ and $v_{\theta_0}(\boldsymbol{x},t_0) = \psi_2(\boldsymbol{x})$. For the exact solution (u,v), we have $R_{int1_i}[u,v] = R_{int2_i}[u,v] = R_{tb1_i}[u] = R_{tb2_i}[v] = R_{sb1_i}[v] = R_{sb2_i}[u] = 0$.

With the HLConcPINN algorithm, we minimize the quantity (for $1 \le i \le l$),

$$\mathcal{E}_{G_i}(\theta_i)^2 = \widetilde{\mathcal{E}}_{G_i}(\theta_i)^2 + \mathcal{E}_{G_{i-1}}(\theta_{i-1})^2, \tag{38}$$

$$\begin{aligned}\widetilde{\mathcal{E}}_{G_i}(\theta_i)^2 = &\int_{\Omega_i} (|R_{int1_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t)|^2 + |R_{int2_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t)|^2 + |\nabla R_{int1_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t)|^2)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \\ &+ \sum_{j=1}^{i} \int_D (|R_{tb1_i}[u_{\theta_i}](\boldsymbol{x},t_{j-1})|^2 + |R_{tb2_i}[v_{\theta_i}](\boldsymbol{x},t_{j-1})|^2 + |\nabla R_{tb1_i}[u_{\theta_i}](\boldsymbol{x},t_{j-1})|^2)\,\mathrm{d}\boldsymbol{x} \\ &+ \left(\int_{\Omega_{*i}} |R_{sb1_i}[v_{\theta_i}](\boldsymbol{x},t)|^2\,\mathrm{d}s(\boldsymbol{x})\mathrm{d}t \right)^{\frac{1}{2}} + \left(\int_{\Omega_{*i}} |R_{sb2_i}[u_{\theta_i}](\boldsymbol{x},t)|^2\,\mathrm{d}s(\boldsymbol{x})\mathrm{d}t \right)^{\frac{1}{2}}.\end{aligned} \tag{39}$$

Here, $\mathcal{E}_{G_0}(\theta_0) = 0$.

Adopting the full training set $\mathcal{S} = \bigcup_{i=1}^{l} \mathcal{S}_i$ with $\mathcal{S}_i = \mathcal{S}_{int_i} \cup \mathcal{S}_{sb_i} \cup \mathcal{S}_{tb_i}$ as given in Section 2.5, we approximate the integrals in (38) by the midpoint rule, resulting in the training loss,

$$\mathcal{E}_{T_i}(\theta_i,\mathcal{S}_i)^2 = \widetilde{\mathcal{E}}_{T_i}(\theta_i,\mathcal{S}_i)^2 + \mathcal{E}_{T_{i-1}}(\theta_{i-1},\mathcal{S}_{i-1})^2, \tag{40}$$

$$\begin{aligned}\widetilde{\mathcal{E}}_{T_i}(\theta_i,\mathcal{S}_i)^2 = &\mathcal{E}_T^{int1_i}(\theta_i,\mathcal{S}_{int_i})^2 + \mathcal{E}_T^{int2_i}(\theta_i,\mathcal{S}_{int_i})^2 + \mathcal{E}_T^{int3_i}(\theta_i,\mathcal{S}_{int_i})^2 + \mathcal{E}_T^{tb1_i}(\theta_i,\mathcal{S}_{tb_i})^2 \\ &+ \mathcal{E}_T^{tb2_i}(\theta_i,\mathcal{S}_{tb_i})^2 + \mathcal{E}_T^{tb3_i}(\theta_i,\mathcal{S}_{tb_i})^2 + \mathcal{E}_T^{sb1_i}(\theta_i,\mathcal{S}_{sb_i}) + \mathcal{E}_T^{sb2_i}(\theta_i,\mathcal{S}_{sb_i}),\end{aligned} \tag{41}$$

where

$$\mathcal{E}_T^{int1_i}(\theta_i,\mathcal{S}_{int_i})^2 = \sum_{n=1}^{N_{int_i}} \omega_{int_i}^n |R_{int1_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x}_{int_i}^n,t_{int_i}^n)|^2, \tag{42a}$$

$$\mathcal{E}_T^{int2_i}(\theta_i,\mathcal{S}_{int_i})^2 = \sum_{n=1}^{N_{int_i}} \omega_{int_i}^n |R_{int2_i}[u_{\theta_i},v_{\theta_i}]](\boldsymbol{x}_{int_i}^n,t_{int_i}^n)|^2, \tag{42b}$$

$$\mathcal{E}_T^{int3_i}(\theta_i,\mathcal{S}_{int_i})^2 = \sum_{n=1}^{N_{int_i}} \omega_{int_i}^n |\nabla R_{int1_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x}_{int_i}^n,t_{int_i}^n)|^2, \tag{42c}$$

$$\mathcal{E}_T^{tb1_i}(\theta_i,\mathcal{S}_{tb_i})^2 = \sum_{j=1}^{i} \sum_{n=1}^{N_{tb_i}} \omega_{tb_i}^n |R_{tb1_i}[u_{\theta_i}](\boldsymbol{x}_{tb_i}^n,t_{j-1})|^2, \tag{42d}$$

$$\mathcal{E}_T^{tb2_i}(\theta_i,\mathcal{S}_{tb_i})^2 = \sum_{j=1}^{i} \sum_{n=1}^{N_{tb_i}} \omega_{tb_i}^n |R_{tb2_i}[v_{\theta_i}](\boldsymbol{x}_{tb_i}^n,t_{j-1})|^2, \tag{42e}$$

$$\mathcal{E}_T^{tb3_i}(\theta_i,\mathcal{S}_{tb_i})^2 = \sum_{j=1}^{i} \sum_{n=1}^{N_{tb_i}} \omega_{tb_i}^n |\nabla R_{tb1_i}[u_{\theta_i}](\boldsymbol{x}_{tb_i}^n,t_{j-1})|^2, \tag{42f}$$

$$\mathcal{E}_T^{sb1_i}(\theta_i, S_{sb_i})^2 = \sum_{n=1}^{N_{sb_i}} \omega_{sb_i}^n |R_{sb1_i}[v_{\theta_i}](\boldsymbol{x}_{sb_i}^n, t_{sb_i}^n)|^2, \tag{42g}$$

$$\mathcal{E}_T^{sb2_i}(\theta_i, S_{sb_i})^2 = \sum_{n=1}^{N_{sb_i}} \omega_{sb_i}^n |R_{sb2_i}[u_{\theta_i}](\boldsymbol{x}_{sb_i}^n, t_{sb_i}^n)|^2. \tag{42h}$$

Here, the quadrature points in space-time constitute the data sets $S_{int_i} = \{(\boldsymbol{x}_{int_i}^n, t_{int_i}^n)\}_{n=1}^{N_{int_i}}$, $S_{tb_i} = \{\boldsymbol{x}_{tb_i}^n\}_{n=1}^{N_{tb_i}}$ and $S_{sb_i} = \{(\boldsymbol{x}_{sb_i}^n, t_{sb_i}^n)\}_{n=1}^{N_{sb_i}}$, and $\omega_{\star_i}^n$ are suitable quadrature weights with $\star$ denoting int, tb or sb. Notice that $\mathcal{E}_{T_{i-1}}(\theta_{i-1}, S_{i-1}) = 0$ for $i = 1$.

5.3. Error analysis

Let $\hat{u}_i = u_{\theta_i} - u$, $\hat{v}_i = v_{\theta_i} - v$ denote the error of the HLConcPINN approximation of the solution (u, v). We define the total approximation error by

$$\mathcal{E}(\theta_i)^2 = \int_{t_{i-1}}^{t_i} \int_D (|\hat{u}_i(\boldsymbol{x}, t)|^2 + |\nabla \hat{u}_i(\boldsymbol{x}, t)|^2 + |\hat{v}_i(\boldsymbol{x}, t)|^2) \mathrm{d}\boldsymbol{x}\, \mathrm{d}t. \tag{43}$$

In light of the wave equations (36) and the definitions of residuals (37), we have

$$R_{int1_i} = \frac{\partial \hat{u}_i}{\partial t} - \hat{v}_i, \tag{44a}$$

$$R_{int2_i} = \frac{\partial \hat{v}_i}{\partial t} - \Delta \hat{u}_i, \tag{44b}$$

$$R_{tb1_i}|_{t=t_{j-1}} = \hat{u}_i|_{t=t_{j-1}} - \hat{u}_{j-1}|_{t=t_{j-1}} \qquad j = 1, 2, \cdots, i, \tag{44c}$$

$$R_{tb2_i}|_{t=t_{j-1}} = \hat{v}_i|_{t=t_{j-1}} - \hat{v}_{j-1}|_{t=t_{j-1}} \qquad j = 1, 2, \cdots, i, \tag{44d}$$

$$R_{sb1_i} = \hat{v}_i(\boldsymbol{x}, t) - \hat{v}_i(\boldsymbol{x} + 1, t), \quad R_{sb2_i} = \nabla \hat{u}_i(\boldsymbol{x}, t) - \nabla \hat{u}_i(\boldsymbol{x} + 1, t). \tag{44e}$$

Theorem 5.1. *Let $\widetilde{\Omega}_i = D \times [0, t_i]$ and $\widetilde{\Omega}_{*i} = \partial D \times [0, t_i]$ $(1 \le i \le l)$. Let n, d, $k \in \mathbb{N}$ with $n \ge 2$ and $k \ge 3$, $u \in H^k(\widetilde{\Omega}_i)$ and $v \in H^{k-1}(\widetilde{\Omega}_i)$. For every integer $N > 5$ and $1 \le j \le i \le l$, there exist HLConcPINNs u_{θ_i} and v_{θ_i}, such that*

$$\|R_{int1_i}\|_{L^2(\widetilde{\Omega}_i)}, \|R_{tb1_i}(\boldsymbol{x}, t_{j-1})\|_{L^2(D)} \lesssim N^{-k+1} \ln N, \tag{45a}$$

$$\|R_{int2_i}\|_{L^2(\widetilde{\Omega}_i)}, \|\nabla R_{int1_i}\|_{L^2(\widetilde{\Omega}_i)}, \|\nabla R_{tb1_i}(\boldsymbol{x}, t_{j-1})\|_{L^2(D)}, \|R_{sb2_i}\|_{L^2(\widetilde{\Omega}_{*i})} \lesssim N^{-k+2} \ln^2 N, \tag{45b}$$

$$\|R_{tb2_i}(\boldsymbol{x}, t_{j-1})\|_{L^2(D)}, \|R_{sb1_i}\|_{L^2(\widetilde{\Omega}_{*i})} \lesssim N^{-k+2} \ln N. \tag{45c}$$

Proof. Similar to Theorem 3.3, the proof follows by noting $u \in H^k(\widetilde{\Omega}_i)$, $v \in H^{k-1}(\widetilde{\Omega}_i)$, Lemmas 9.3 and 9.8. □

Theorem 5.1 implies that one can make the HLConcPINN residuals (37) arbitrarily small by choosing N to be sufficiently large. It follows that the generalization error $\mathcal{E}_{G_i}(\theta_i)^2$ in (38) can be made arbitrarily small. The next two theorems show that: (i) the total approximation error $\mathcal{E}(\theta_i)^2$ is small when the generalization error $\mathcal{E}_{G_i}(\theta_i)^2$ is small with the HLConcPINN approximation $(u_{\theta_i}, v_{\theta_i})$, and (ii) the total approximation error $\mathcal{E}(\theta_i)^2$ can be arbitrarily small if the training error $\mathcal{E}_{T_i}(\theta_i, S_i)^2$ is sufficiently small and the sample set is sufficiently large.

Theorem 5.2. *Let $d \in \mathbb{N}$, $u \in C^1(\widetilde{\Omega}_i)$ and $v \in C^0(\widetilde{\Omega}_i)$ be the classical solution to (36). Let u_{θ_i} and v_{θ_i} denote the HLConcPINN approximation with parameter θ_i. For all $1 \le i \le l$, the following relation holds,*

$$\int_D (|\hat{u}_i(\boldsymbol{x}, \tau)|^2 + |\nabla \hat{u}_i(\boldsymbol{x}, \tau)|^2 + |\hat{v}_i(\boldsymbol{x}, \tau)|^2) \mathrm{d}\boldsymbol{x} \le C_{G_i} \exp(2\Delta t) \quad \tau \in [t_{i-1}, t_i], \tag{46}$$

$$\int_{t_{i-1}}^{t_i} \int_D (|\hat{u}_i(\boldsymbol{x}, t)|^2 + |\nabla \hat{u}_i(\boldsymbol{x}, t)|^2 + |\hat{v}_i(\boldsymbol{x}, t)|^2) \mathrm{d}\boldsymbol{x}\, \mathrm{d}t \le C_{G_i} \Delta t \exp(2\Delta t), \tag{47}$$

where $\Delta t = t_i - t_{i-1}$ and for $1 \le i \le l$,

$$C_{G_i} = \widetilde{C}_{G_i} + 2C_{G_{i-1}} \exp(2\Delta t), \qquad C_{G_0} = 0,$$

$$\widetilde{C}_{G_i} = \int_{\widetilde{\Omega}_i} (|R_{int1_i}|^2 + |R_{int2_i}|^2 + |\nabla R_{int1_i}|^2) \mathrm{d}\boldsymbol{x}\, \mathrm{d}t$$

$$+2\sum_{j=1}^{i}\int_{D}(|R_{tb1_i}(\boldsymbol{x},t_{j-1})|^2+|R_{tb2_i}(\boldsymbol{x},t_{j-1})|^2+|\nabla R_{tb1_i}(\boldsymbol{x},t_{j-1})|^2)\mathrm{d}\boldsymbol{x}$$

$$+2|\Delta t|^{\frac{1}{2}}C_{\partial D_{1i}}\left(\int_{\Omega_{*i}}|R_{sb1_i}|^2\mathrm{d}s(\boldsymbol{x})\mathrm{d}t\right)^{\frac{1}{2}}+2|\Delta t|^{\frac{1}{2}}C_{\partial D_{2i}}\left(\int_{\Omega_{*i}}|R_{sb2_i}|^2\mathrm{d}s(\boldsymbol{x})\mathrm{d}t\right)^{\frac{1}{2}},$$

$C_{\partial D_{1i}}=|\partial D|^{\frac{1}{2}}(\|u\|_{C^1(\Omega_{*i})}+\|u_{\theta_i}\|_{C^1(\Omega_{*i})})$ *and* $C_{\partial D_{2i}}=|\partial D|^{\frac{1}{2}}(\|v\|_{C(\Omega_{*i})}+\|v_{\theta_i}\|_{C(\Omega_{*i})})$.

Proof. The proof follows the same techniques as in the proof of Theorem 3.4 and in the proof of Theorem 3.4 of [41]. □

Using the midpoint rule $\mathcal{Q}_M^{\Lambda}[f]$ as described in Section 2.2, we can express the training loss function (41) as

$$\begin{aligned}\widetilde{\mathcal{E}}_{T_i}(\theta_i,S_i)^2&=\mathcal{E}_T^{int1_i}(\theta_i,S_{int_i})^2+\mathcal{E}_T^{int2_i}(\theta_i,S_{int_i})^2+\mathcal{E}_T^{int3_i}(\theta_i,S_{int_i})^2+\mathcal{E}_T^{sb1_i}(\theta_i,S_{sb_i})+\mathcal{E}_T^{sb2_i}(\theta_i,S_{sb_i})\\&\quad+\mathcal{E}_T^{tb1_i}(\theta_i,S_{tb_i})^2+\mathcal{E}_T^{tb2_i}(\theta_i,S_{tb_i})^2+\mathcal{E}_T^{tb3_i}(\theta_i,S_{tb_i})^2,\\&=\mathcal{Q}^{\Omega_i}_{M_{int_i}}[R^2_{int1_i}]+\mathcal{Q}^{\Omega_i}_{M_{int_i}}[R^2_{int2_i}]+\mathcal{Q}^{\Omega_i}_{M_{int_i}}[|\nabla R_{int1_i}|^2]+(\mathcal{Q}^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb1_i}])^{\frac{1}{2}}+(\mathcal{Q}^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb2_i}])^{\frac{1}{2}}\\&\quad+\sum_{j=1}^{i}\left(\mathcal{Q}^{D}_{M_{tb_i}}[R^2_{tb1_i}(\boldsymbol{x},t_{j-1})]+\mathcal{Q}^{D}_{M_{tb_i}}[R^2_{tb2_i}(\boldsymbol{x},t_{j-1})]+\mathcal{Q}^{D}_{M_{tb_i}}[|\nabla R_{tb1_i}(\boldsymbol{x},t_{j-1})|^2]\right).\end{aligned}$$

Then by applying Theorem 5.2, we obtain the following result, which bounds the total error of HLConcPINN in terms of the training loss and the number of training points.

Theorem 5.3. *Let $d\in\mathbb{N}$ and $T>0$. Let $u\in C^4(\widetilde{\Omega}_i)$ and $v\in C^3(\widetilde{\Omega}_i)$ be the classical solution to (36), and let $(u_{\theta_i},v_{\theta_i})$ denote the HLConcPINN approximation with parameter θ_i. Then the total approximation error satisfies*

$$\begin{aligned}&\int_{t_{i-1}}^{t_i}\int_{D}(|\hat{u}_i(\boldsymbol{x},t)|^2+|\nabla\hat{u}_i(\boldsymbol{x},t)|^2+|\hat{v}_i(\boldsymbol{x},t)|^2)\mathrm{d}\boldsymbol{x}\mathrm{d}t\le C_{T_i}\Delta t\exp(2\Delta t)\\&\quad=\mathcal{O}(\mathcal{E}_{T_i}(\theta_i,S_i)^2+M_{int_i}^{-\frac{2}{d+1}}+M_{tb_i}^{-\frac{2}{d}}+M_{sb_i}^{-\frac{1}{d}}),\end{aligned}\tag{48}$$

where

$$\begin{aligned}C_{T_i}&=\widetilde{C}_{T_i}+2C_{T_{i-1}}\exp(2\Delta t),\qquad C_{T_0}=0,\\\widetilde{C}_{T_i}&=2\sum_{j=1}^{i}\left(C_{(R^2_{tb1_i}(\boldsymbol{x},t_{j-1}))}M_{tb_i}^{-\frac{2}{d}}+\mathcal{Q}^{D}_{M_{tb_i}}[R^2_{tb1_i}(\boldsymbol{x},t_{j-1})]+C_{(R^2_{tb2_i}(\boldsymbol{x},t_{j-1}))}M_{tb_i}^{-\frac{2}{d}}+\mathcal{Q}^{D}_{M_{tb_i}}[R^2_{tb2_i}(\boldsymbol{x},t_{j-1})]\right)\\&\quad+2\sum_{j=1}^{i}\left(C_{(|\nabla R_{tb1_i}(\boldsymbol{x},t_{j-1})|^2)}M_{tb_i}^{-\frac{2}{d}}+\mathcal{Q}^{D}_{M_{tb_i}}[|\nabla R_{tb1_i}(\boldsymbol{x},t_{j-1})|^2]\right)+C_{(R^2_{int1_i})}M_{int_i}^{-\frac{2}{d+1}}+\mathcal{Q}^{\Omega_i}_{M_{int_i}}[R^2_{int1_i}]\\&\quad+C_{(R^2_{int2_i})}M_{int_i}^{-\frac{2}{d+1}}+\mathcal{Q}^{\Omega_i}_{M_{int_i}}[R^2_{int2_i}]+C_{(|\nabla R_{int1_i}|^2)}M_{int_i}^{-\frac{2}{d+1}}+\mathcal{Q}^{\Omega_i}_{M_{int_i}}[|\nabla R_{int1_i}|^2]\\&\quad+2|\Delta t|^{\frac{1}{2}}\left(C_{\partial D_{1i}}(C_{(R^2_{sb1_i})}M_{sb_i}^{-\frac{2}{d}}+\mathcal{Q}^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb1_i}])^{\frac{1}{2}}+C_{\partial D_{2i}}(C_{(R^2_{sb2_i})}M_{sb_i}^{-\frac{2}{d}}+\mathcal{Q}^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb2_i}])^{\frac{1}{2}}\right).\end{aligned}\tag{49}$$

Proof. Using Lemma 9.3, Theorem 5.2 and the quadrature error formula (3) leads to this result. □

6. HLConcPINN for approximating the nonlinear Klein-Gordon equation

6.1. Nonlinear Klein-Gordon equation

Let $D\subset\mathbb{R}^d$ be an open connected bounded set with boundary ∂D. We consider the nonlinear Klein-Gordon equation:

$$\frac{\partial u(\boldsymbol{x},t)}{\partial t}-v(\boldsymbol{x},t)=0\qquad(\boldsymbol{x},t)\in D\times[0,T],\tag{50a}$$

$$\varepsilon^2\frac{\partial v(\boldsymbol{x},t)}{\partial t}=a^2\Delta u(\boldsymbol{x},t)-\varepsilon_1^2u(\boldsymbol{x},t)-g(u(\boldsymbol{x},t))+f(\boldsymbol{x},t)\qquad(\boldsymbol{x},t)\in D\times[0,T],\tag{50b}$$

$$u(\boldsymbol{x},0)=\psi_1(\boldsymbol{x})\qquad\boldsymbol{x}\in D,\tag{50c}$$

$$v(\boldsymbol{x},0)=\psi_2(\boldsymbol{x}) \qquad \boldsymbol{x}\in D, \tag{50d}$$

$$u(\boldsymbol{x},t)=u_d(\boldsymbol{x},t) \qquad (\boldsymbol{x},t)\in \partial D\times[0,T], \tag{50e}$$

where u and v are the field functions to be solved for, f is a source term, and u_d, ψ_1 and ψ_2 denote the boundary/initial conditions. $\varepsilon>0$, $a>0$ and $\varepsilon_1\geq 0$ are constants. $g(u)$ is a nonlinear term. We assume that g is globally Lipschitz, i.e. there exists a constant L (independent of v and w) such that

$$|g(v)-g(w)|\leq L|v-w| \qquad \forall v,\, w\in\mathbb{R}. \tag{51}$$

6.2. Hidden-layer concatenated physics informed neural networks

Following the settings in Section 2.5, we define the following residuals for the HLConcPINN approximation $u_{\theta_i}: D\times[0,t_i]\to\mathbb{R}$ and $v_{\theta_i}: D\times[0,t_i]\to\mathbb{R}$ (for $1\leq j\leq i\leq l$) of the equations in (50):

$$R_{int1_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t)=\frac{\partial u_{\theta_i}}{\partial t}-v_{\theta_i}, \quad R_{int2_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t)=\varepsilon^2\frac{\partial v_{\theta_i}}{\partial t}-a^2\Delta u_{\theta_i}+\varepsilon_1^2 u_{\theta_i}+g(u_{\theta_i})-f, \tag{52a}$$

$$R_{tb1_i}[u_{\theta_i}](\boldsymbol{x},t_{j-1})=u_{\theta_i}(\boldsymbol{x},t_{j-1})-u_{\theta_{j-1}}(\boldsymbol{x},t_{j-1}), \quad R_{tb2_i}[v_{\theta_i}](\boldsymbol{x},t_{j-1})=v_{\theta_i}(\boldsymbol{x},t_{j-1})-v_{\theta_{j-1}}(\boldsymbol{x},t_{j-1}), \tag{52b}$$

$$R_{sb_i}[v_{\theta_i}](\boldsymbol{x},t)=v_{\theta_i}(\boldsymbol{x},t)|_{\partial D}-u_{dt}(\boldsymbol{x},t), \tag{52c}$$

where $u_{dt}=\frac{\partial u_d}{\partial t}$, $u_{\theta_0}(\boldsymbol{x},t_0)=\psi_1(\boldsymbol{x})$ and $v_{\theta_0}(\boldsymbol{x},t_0)=\psi_2(\boldsymbol{x})$. Notice that $R_{int1_i}[u,v]=R_{int2_i}[u,v]=R_{tb1_i}[u]=R_{tb2_i}[v]=R_{sb_i}[v]=0$ for the exact solution (u,v). We minimize the following generalization error (for $1\leq i\leq l$),

$$\mathcal{E}_{G_i}(\theta_i)^2=\widetilde{\mathcal{E}}_{G_i}(\theta_i)^2+\mathcal{E}_{G_{i-1}}(\theta_{i-1})^2, \tag{53}$$

$$\begin{aligned}\widetilde{\mathcal{E}}_{G_i}(\theta_i)^2=&\int_{\Omega_i}\left(|R_{int1_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t)|^2+|R_{int2_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t)|^2+|\nabla R_{int1_i}[u_{\theta_i},v_{\theta_i}](\boldsymbol{x},t)|^2\right)\mathrm{d}\boldsymbol{x}\,\mathrm{d}t\\&+\sum_{j=1}^{i}\int_D\left(|R_{tb1_i}[u_{\theta_i}](\boldsymbol{x},t_{j-1})|^2+|R_{tb2_i}[v_{\theta_i}](\boldsymbol{x},t_{j-1})|^2+|\nabla R_{tb1_i}[u_{\theta_i}](\boldsymbol{x},t_{j-1})|^2\right)\mathrm{d}\boldsymbol{x}\\&+\left(\int_{\Omega_{*i}}|R_{sb_i}[v_{\theta_i}](\boldsymbol{x},t)|^2\,\mathrm{d}s(\boldsymbol{x})\mathrm{d}t\right)^{\frac{1}{2}},\end{aligned} \tag{54}$$

where $\mathcal{E}_{G_0}(\theta_0)=0$.

Employing the training set $\mathcal{S}=\bigcup_{i=1}^{l}\mathcal{S}_i$ with $\mathcal{S}_i=\mathcal{S}_{int_i}\cup\mathcal{S}_{sb_i}\cup\mathcal{S}_{tb_i}$ as given in Section 2.5 and the midpoint rule for approximating the residuals, we arrive at the training loss as follows (for $1\leq i\leq l$),

$$\mathcal{E}_{T_i}(\theta_i,\mathcal{S}_i)^2=\widetilde{\mathcal{E}}_{T_i}(\theta_i,\mathcal{S}_i)^2+\mathcal{E}_{T_{i-1}}(\theta_{i-1},\mathcal{S}_{i-1})^2, \tag{55}$$

$$\begin{aligned}\widetilde{\mathcal{E}}_{T_i}(\theta_i,\mathcal{S}_i)^2=&\mathcal{E}_T^{int1_i}(\theta_i,\mathcal{S}_{int_i})^2+\mathcal{E}_T^{int2_i}(\theta_i,\mathcal{S}_{int_i})^2+\mathcal{E}_T^{int3_i}(\theta_i,\mathcal{S}_{int_i})^2+\mathcal{E}_T^{tb1_i}(\theta_i,\mathcal{S}_{tb_i})^2\\&+\mathcal{E}_T^{tb2_i}(\theta_i,\mathcal{S}_{tb_i})^2+\mathcal{E}_T^{tb3_i}(\theta_i,\mathcal{S}_{tb_i})^2+\mathcal{E}_T^{sb_i}(\theta_i,\mathcal{S}_{sb_i}),\end{aligned} \tag{56}$$

where $\mathcal{E}_T^{sb_i}(\theta_i,\mathcal{S}_{sb_i})^2=\sum_{n=1}^{N_{sb_i}}\omega_{sb_i}^n|R_{sb_i}[v_{\theta_i}](\boldsymbol{x}_{sb_i}^n,t_{sb_i}^n)|^2$ and the other terms are defined in (42). Notice that $\mathcal{E}_{T_0}(\theta_0,\mathcal{S}_0)=0$.

Remark 6.1. The parameter ε can influence the stiffness of the problem and the practical difficulty in solving the problem. When ε is small, the wavelength decreases and the frequency increases. To improve accuracy, one can reduce the size of the time blocks, effectively increasing the number of time blocks to account for the higher frequency.

6.3. Error analysis

Let $\hat{u}_i=u_{\theta_i}-u$ and $\hat{v}_i=v_{\theta_i}-v$ denote the errors of HLConcPINN approximation, where (u,v) are the exact solutions. We define the total approximation error of HLConcPINN as,

$$\mathcal{E}(\theta_i)^2=\int_{t_{i-1}}^{t_i}\int_D(|\hat{u}_i(\boldsymbol{x},t)|^2+a^2|\nabla\hat{u}_i(\boldsymbol{x},t)|^2+\varepsilon^2|\hat{v}_i(\boldsymbol{x},t)|^2)\mathrm{d}\boldsymbol{x}\,\mathrm{d}t. \tag{57}$$

Subtracting the equations (50) from the residual equations (52) leads to,

$$R_{int1_i} = \frac{\partial \hat{u}_i}{\partial t} - \hat{v}_i, \tag{58a}$$

$$R_{int2_i} = \varepsilon^2 \frac{\partial \hat{v}_i}{\partial t} - a^2 \Delta \hat{u}_i + \varepsilon_1^2 \hat{u}_i + g(u_{\theta_i}) - g(u), \tag{58b}$$

$$R_{tb1_i}|_{t=t_{j-1}} = \hat{u}_i|_{t=t_{j-1}} - \hat{u}_{j-1}|_{t=t_{j-1}} \qquad j = 1, 2, \cdots, i, \tag{58c}$$

$$R_{tb2_i}|_{t=t_{j-1}} = \hat{v}_i|_{t=t_{j-1}} - \hat{v}_{j-1}|_{t=t_{j-1}} \qquad j = 1, 2, \cdots, i, \tag{58d}$$

$$R_{sb_i} = \hat{v}_i|_{\partial D}. \tag{58e}$$

The following theorems summarize the results of the HLConcPINN approximation for the nonlinear Klein-Gordon equation.

Theorem 6.2. *Let $n \geq 2$, d, $k \in \mathbb{N}$ with $k \geq 3$. Suppose that $g(u)$ is Lipschitz continuous, $u \in C^k(D \times [0, t_i])$ and $v \in C^{k-1}(D \times [0, t_i])$ $(1 \leq i \leq l)$. Then for every integer $N > 5$, there exist HLConcPINNs u_{θ_i} and v_{θ_i}, such that*

$$\|R_{int1_i}\|_{L^2(D\times[0,t_i])}, \|R_{tb1_i}\|_{L^2(D)} \lesssim N^{-k+1} \ln N, \tag{59a}$$

$$\|R_{int2_i}\|_{L^2(D\times[0,t_i])}, \|\nabla R_{int1_i}\|_{L^2(D\times[0,t_i])}, \|\nabla R_{tb1_i}\|_{L^2(D)} \lesssim N^{-k+2} \ln^2 N, \tag{59b}$$

$$\|R_{tb2_i}\|_{L^2(D)}, \|R_{sb_i}\|_{L^2(\partial D\times[0,t_i])} \lesssim N^{-k+2} \ln N. \tag{59c}$$

Proof. Similar to that of Theorem 3.3, the proof follows by noting $u \in C^k(D \times [0, t_i])$, $v \in C^{k-1}(D \times [0, t_i])$, Lemmas 9.3, 9.5 and 9.8, and the globally Lipschitz condition (51). □

This theorem implies that the HLConcPINN residuals in (52) can be made arbitrarily small by choosing a sufficiently large N. Therefore, the generalization error $\mathcal{E}_{G_i}(\theta_i)^2$ can be made arbitrarily small. We next show that the HLConcPINN total approximation error $\mathcal{E}(\theta_i)^2$ can be controlled by the generalization error $\mathcal{E}_{G_i}(\theta_i)^2$ (Theorem 6.3 below), and by the training error $\mathcal{E}_{T_i}(\theta_i, S_i)^2$ (Theorem 6.4 below).

Theorem 6.3. *Let $d \in \mathbb{N}$, $u \in C^1(D \times [0, t_i])$ and $v \in C^0(D \times [0, t_i])$ be the classical solution of (50). Let $(u_{\theta_i}, v_{\theta_i})$ denote the HLConcPINN approximation with parameter θ_i. For $1 \leq i \leq l$, the following relation holds,*

$$\int_D (|\hat{u}_i(\boldsymbol{x}, t)|^2 + a^2 |\nabla \hat{u}_i(\boldsymbol{x}, t)|^2 + \varepsilon^2 |\hat{v}_i(\boldsymbol{x}, t)|^2) \mathrm{d}\boldsymbol{x} \leq C_{G_i} \exp\left((2 + \varepsilon_1^2 + L + a^2)\Delta t\right), \tag{60}$$

$$\int_{t_{i-1}}^{t_i} \int_D (|\hat{u}_i(\boldsymbol{x}, t)|^2 + a^2 |\nabla \hat{u}_i(\boldsymbol{x}, t)|^2 + \varepsilon^2 |\hat{v}_i(\boldsymbol{x}, t)|^2) \mathrm{d}\boldsymbol{x}\mathrm{d}t \leq C_{G_i} \Delta t \exp\left((2 + \varepsilon_1^2 + L + a^2)\Delta t\right), \tag{61}$$

where

$$C_{G_i} = \widetilde{C}_{G_i} + 2C_{G_{i-1}} \exp((2 + \varepsilon_1^2 + L + a^2)\Delta t), \qquad C_{G_0} = 0,$$

$$\widetilde{C}_{G_i} = \int_{\Omega_i} (|R_{int1_i}|^2 + |R_{int2_i}|^2 + a^2 |\nabla R_{int1_i}|^2) \mathrm{d}\boldsymbol{x}\mathrm{d}t + 2C_{\partial D_i} |\Delta t|^{\frac{1}{2}} \left(\int_{\Omega_{*i}} |R_{sb_i}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t \right)^{\frac{1}{2}}$$
$$+ 2 \sum_{j=1}^{i} \int_D (|R_{tb1_i}(\boldsymbol{x}, t_{j-1})|^2 + \varepsilon^2 |R_{tb2_i}(\boldsymbol{x}, t_{j-1})|^2 + a^2 |\nabla R_{tb1_i}(\boldsymbol{x}, t_{j-1})|^2) \mathrm{d}\boldsymbol{x},$$

and $C_{\partial D_i} = a^2 |\partial D|^{\frac{1}{2}} (\|u\|_{C^1(\partial D\times[t_{i-1},t_i])} + \|u_{\theta_i}\|_{C^1(\partial D\times[t_{i-1},t_i])})$.

Proof. The proof is similar to that of Theorem 3.4, by noting (51). □

Theorem 6.4. *Let $d \in \mathbb{N}$ and $T > 0$, and let $u \in C^4(D \times [0, t_i])$ and $v \in C^3(D \times [0, t_i])$ be the classical solution to (50). Let $(u_{\theta_i}, v_{\theta_i})$ denote the HLConcPINN approximation with parameter θ_i. Then the following relation holds $(1 \leq i \leq l)$,*

$$\int_{\Omega_i} (|\hat{u}_i(\boldsymbol{x}, t)|^2 + a^2 |\nabla \hat{u}_i(\boldsymbol{x}, t)|^2 + \varepsilon^2 |\hat{v}_i(\boldsymbol{x}, t)|^2) \mathrm{d}\boldsymbol{x}\mathrm{d}t \leq C_{T_i} \Delta t \exp\left((2 + \varepsilon_1^2 + L + a^2)\Delta t\right)$$
$$= \mathcal{O}(\mathcal{E}_{T_i}(\theta_i, S_i)^2 + M_{int_i}^{-\frac{2}{d+1}} + M_{tb_i}^{-\frac{2}{d}} + M_{sb_i}^{-\frac{1}{d}}), \tag{62}$$

where

Table 1
Summary of neural network settings (network architecture, activation functions, training data points) used within each time block for the test problems in Section 7. Shown in the second column are the nodes in the hidden layers only. This configuration is applied consistently across all time blocks.

N_c	Hidden layers	Activation functions	Figures/Tables
varied	[90, 90]	[tanh, tanh]	Table 2
	[90, 90, 10]	[tanh, tanh, sine]	Tables 5, 8, 10
2000	varied	all tanh	Table 3
	[90, 90]	[tanh, tanh]	Fig. 6a
	[90, 90, 10]	[tanh, tanh, tanh]	Figs. 3–5, 6b
	[90, 90, 10, 10]	[tanh, tanh, tanh, tanh]	Table 4
		[tanh, tanh, Gaussian, Gaussian]	Fig. 6c
		[tanh, tanh, softplus, softplus]	Fig. 6d
2500	[90, 90, 10]	varied	Tables 6, 9, 11
	[90, 90, 10]	[tanh, tanh, sine]	Figs. 7–18

$$
\begin{aligned}
C_{T_i} &= \widetilde{C}_{T_i} + 2C_{T_{i-1}} \exp\left((2+\varepsilon_1^2+L+a^2)\Delta t\right), \qquad C_{T_0} = 0,\\
\widetilde{C}_{T_i} &= 2\sum_{j=1}^{i}\left(C_{(R^2_{tb1_i}(\boldsymbol{x},t_{j-1}))} M_{tb_i}^{-\frac{2}{d}} + \mathcal{Q}^D_{M_{tb_i}}[R^2_{tb1_i}(\boldsymbol{x},t_{j-1})] + \varepsilon^2 C_{(R^2_{tb2_i}(\boldsymbol{x},t_{j-1}))} M_{tb_i}^{-\frac{2}{d}} + \varepsilon^2 \mathcal{Q}^D_{M_{tb_i}}[R^2_{tb2_i}(\boldsymbol{x},t_{j-1})]\right)\\
&\quad + 2a^2\sum_{j=1}^{i}\left(C_{(|\nabla R_{tb1_i}(\boldsymbol{x},t_{j-1})|^2)} M_{tb_i}^{-\frac{2}{d}} + \mathcal{Q}^D_{M_{tb_i}}[|\nabla R_{tb1_i}(\boldsymbol{x},t_{j-1})|^2]\right) + C_{(R^2_{int1_i})} M_{int_i}^{-\frac{2}{d+1}} + \mathcal{Q}^{\Omega_i}_{M_{int_i}}[R^2_{int1_i}]\\
&\quad + C_{(R^2_{int2_i})} M_{int}^{-\frac{2}{d+1}} + \mathcal{Q}^{\Omega_i}_{M_{int}}[R^2_{int2_i}] + a^2\left(C_{(|\nabla R_{int1_i}|^2)} M_{int}^{-\frac{2}{d+1}} + \mathcal{Q}^{\Omega_i}_{M_{int}}[|\nabla R_{int1_i}|^2]\right),\\
&\quad + 2C_{\partial D_i}|\Delta t|^{\frac{1}{2}}\left(C_{(R^2_{sb_i})} M_{sb}^{-\frac{2}{d}} + \mathcal{Q}^{\Omega_{*i}}_{M_{sb}}[R^2_{sb_i}]\right)^{\frac{1}{2}}.
\end{aligned}
$$

Here, $\mathcal{Q}^{\Lambda}_{M}[f]$ represents the corresponding quadrature for different functions ($f = R^2_{tb1_i}$, $R^2_{tb2_i}$, $|\nabla R_{tb1_i}|^2$, $R^2_{int1_i}$, $R^2_{int2_i}$, $|\nabla R_{int1_i}|^2$ and $R^2_{sb_i}$) and the numbers of quadrature points ($M = M_{int_i}$, M_{sb_i} and M_{tb_i}) with respect to the relevant domain ($\Lambda = \Omega_i$, Ω_{*i} and D).

Proof. The proof follows from Lemma 9.3, Theorem 6.3 and the quadrature error formula (3). □

It follows from Theorem 6.4 that the HLConcPINN approximation error $\mathcal{E}(\theta_i)^2$ can be arbitrarily small, provided that the training error $\mathcal{E}_{T_i}(\theta_i, S_i)^2$ is sufficiently small and the sample set is sufficiently large.

7. Computational examples

We next present a set of numerical examples to test the performance of the HLConcPINN method developed herein. This method has several distinctive features, distinguishing it from the standard PINN and recent neural networks with theoretical guarantees. Specifically, these include:

- The method is based on hidden-layer concatenated FNNs (HLConcFNN), in which the output nodes and all the hidden nodes are logically connected. This architecture is critical to the theoretical analyses, and it endows the method with the subsequent properties.
- The current error analyses hold for network architectures with two or more hidden layers, and with essentially any activation function having a sufficient regularity for all hidden layers beyond the first two. This generalized capability contrasts starkly with the recent PINN methods that have a theoretical guarantee for solving PDEs but are confined to network architectures having two hidden layers and the tanh activation function (see e.g. [12,41]).
- The method espouses a modified block time marching (ExBTM) strategy for long-time dynamic simulations. In the modified scheme, the "initial condition" for a particular time block is informed by the approximations from all previous time blocks evaluated at a set of discrete time instants. The modified BTM scheme is crucial for the error analyses. In contrast, the original BTM formulation as given in e.g. [14] is not amenable to theoretical analysis.

We consider the heat, Burgers', wave and the nonlinear Klein-Gordon equations in one spatial dimension plus time. For each problem, the algorithm involves the following procedure:

- Divide the space-time domain of the problem in time into a number of blocks.

- Choose a set of random collocation points on the interior, the spatial boundaries, and the initial boundary of each time block for network training.
- Loop over the time blocks successively, and on each time block:

  (a) Construct a HLConcFNN with a prescribed architecture and prescribed activation functions.
  (b) Formulate the loss function based on the forms from theoretical analyses of Sections 3–6, and compute the loss over appropriate collocation points.
  (c) Train the HLConcPINN by minimizing the loss function using the combined Adam and L-BFGS optimizers.

The following are the common settings in the numerical tests. We partition the temporal dimension into five uniform time blocks. Within each time block, we utilize N_c collocation points sampled from a uniform random distribution within the spatial-temporal domain. Additionally, N_c uniform random points are selected along each spatial boundary and the initial boundary. Simulations were performed by systematically varying N_c between 1500 and 3000. After the neural networks are trained, we compare the HLConcPINN solution with the exact solution on a set of $N_{ev} = 1000 \times 1000$ uniform grid points (test/evaluation points) within each time block that encompasses the problem domain and its boundaries.

The HLConcPINN errors reported below have been calculated as follows. Suppose $z_n = (\boldsymbol{x}, t)_n \in D \times [0, T]$ $(n = 1, \cdots, N_{ev})$ denote the set of test points. The errors are then defined by

$$l^2\text{-error} = \frac{\sqrt{\sum_{n=1}^{N_{ev}} |u(z_n) - u_\theta(z_n)|^2}}{\sqrt{\sum_{n=1}^{N_{ev}} u(z_n)^2}}, \quad l^\infty\text{-error} = \frac{\max\{|u(z_n) - u_\theta(z_n)|\}_{n=1}^{N_{ev}}}{\sqrt{\left(\sum_{n=1}^{N_{ev}} u(z_n)^2\right)/N_{ev}}}, \tag{63}$$

where u_θ denotes the HLConcPINN solution and u denotes the exact solution.

Following the analyses in previous sections, we employ network architectures with two or more hidden layers in the numerical tests, with the tanh activation function for the first two hidden layers. For the subsequent hidden layers, we have tested a range of activation functions. Table 1 provides an overview of the neural network settings for the test results reported in the subsequent subsections.

As discussed in Remark 3.2, we adopt a modified block time marching scheme in this work. This is different from the original block time marching scheme of [14], which uses the solution data of the preceding time block at the last time instant as the initial condition. Both the original and the modified block time marching schemes have been tested in the simulations, with their results marked by "HLConcPINN-BTM" and "HLConcPINN-ExBTM" in the following discussions, respectively.

In the following simulations, the neural network has been trained by a combination of the Adam optimizer and the L-BFGS optimizer. Within each time block, the network is trained first by Adam for 100 epochs. The training then continues with the L-BFGS optimizer for another 30,000 iterations. Our application code is implemented in Python with the PyTorch library. All the numerical examples are executed on a Ubuntu 22.04 system (3.6 GHz Intel Core i9 CPU, 32 GB memory).

7.1. Heat equation

We test the HLConcPINN scheme for solving the heat equation in one spatial dimension (plus time). Consider the spatial-temporal domain $\Omega = \{(x,t) | x \in [0,1], t \in [0,10]\}$, and the following initial/boundary-value problem,

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} = f(x,t), \tag{64a}$$

$$u(0,t) = g_1(t), \quad u(1,t) = g_2(t), \tag{64b}$$

$$u(x,0) = h(x), \tag{64c}$$

where $u(x,t)$ is the field function to be solved for, $f(x,t)$ is a source term, and $\nu = 0.1$ is the diffusion coefficient. $g_1(x)$ and $g_2(x)$ are the boundary conditions, and $h(x)$ is the initial field distribution. In this test, we choose the source term f such that the following field function satisfies (64),

$$u(x,t) = \left(2\cos(\pi x + \frac{\pi}{5}) + \frac{3}{2}\cos(2\pi x - \frac{3\pi}{5})\right)\left(2\cos(\pi t + \frac{\pi}{5}) + \frac{3}{2}\cos(2\pi t - \frac{3\pi}{5})\right), \tag{65}$$

and we choose the initial/boundary conditions by restricting (65) to the corresponding boundaries.

We consider two forms for the HLConcPINN loss function, corresponding to the original block time marching (BTM) scheme from [14] and the modified BTM scheme (denoted by ExBTM) developed in this work. The loss function for the current ExBTM scheme is given by, for time block i $(1 \le i \le l)$,

$$\begin{aligned} Loss_i^I =& \frac{W_1}{N_c} \sum_{n=1}^{N_c} \left[\frac{\partial u_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - \Delta u_{\theta_i}(x_{int}^n, t_{int}^n) - f(x_{int}^n, t_{int}^n) \right]^2 \\ &+ \frac{W_2}{N_c} \sum_{j=1}^{i} \sum_{n=1}^{N_c} \left[u_{\theta_i}(x_{tb}^n, t_{j-1}) - u_{\theta_{j-1}}(x_{tb}^n, t_{j-1}) \right]^2 \end{aligned}$$

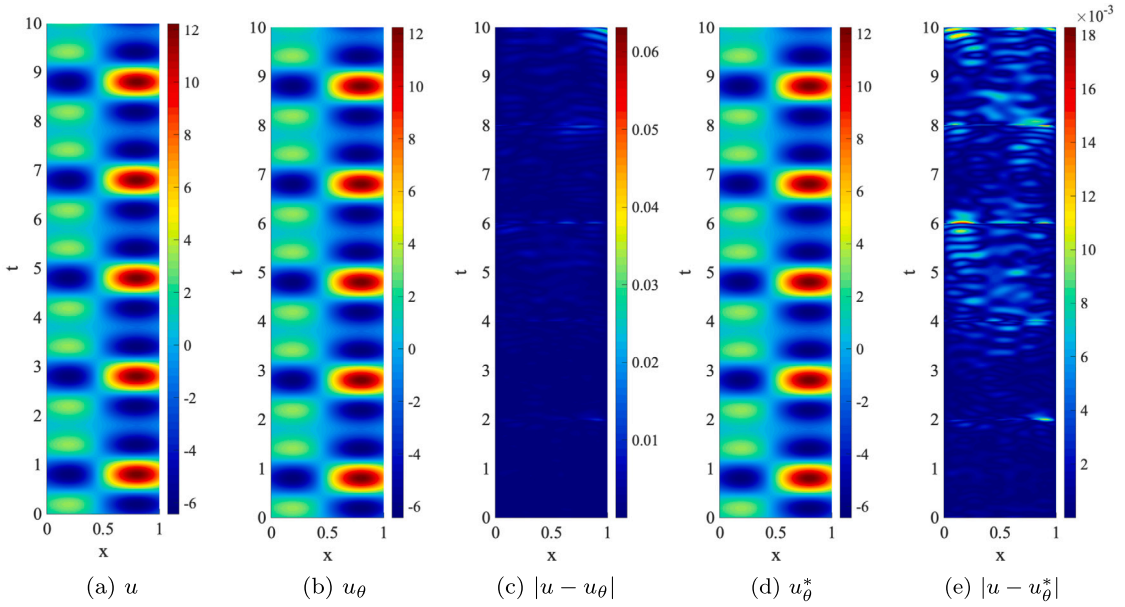


Fig. 3. Heat equation: Distributions of the true solution (a), the HLConcPINN-ExBTM solution (b) and its point-wise absolute error (c), the HLConcPINN-BTM solution (d) (denoted by u^*_θ) and its point-wise absolute error (e), in the spacial-temporal domain. NN architecture: [2, 90, 90, 10, 1], with the tanh activation function; $N_c = 2000$ for the collocation points. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

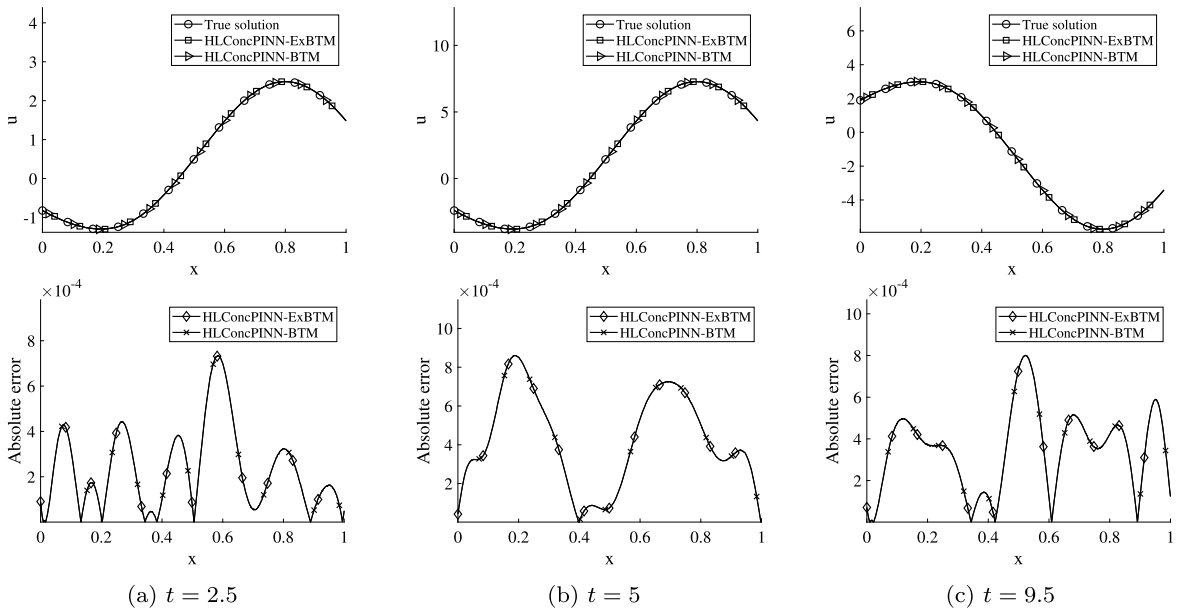


Fig. 4. Heat equation: Top row, comparison of profiles of the true solution, HLConcPINN-ExBTM solution, and HLConcPINN-BTM solution at several time instants. Bottom row, profiles of the absolute error of the HLConcPINN-ExBTM and HLConcPINN-BTM solutions. NN architecture: [2, 90, 90, 10, 1] with tanh activation function; $N_c = 2000$ for the training collocation points.

$$+ W_3\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}\Big[(u_{\theta_i}(0,t^n_{sb}) - g_1(t^n_{sb}))^2 + (u_{\theta_i}(1,t^n_{sb}) - g_2(t^n_{sb}))^2\Big]\Big)^{1/2}$$
$$+ Loss^I_{i-1}, \tag{66}$$

where $Loss^I_0 = 0$, and we have added a set of penalty coefficients $W_k > 0$ $(k = 1, 2, 3)$ for different loss terms. Note also that in the simulations we have approximated the integral by averaging over the collocation points in the domain, while in the analysis the mid-point rule has been adopted. The loss form corresponding to the original BTM scheme is, for time block i $(1 \le i \le l)$,

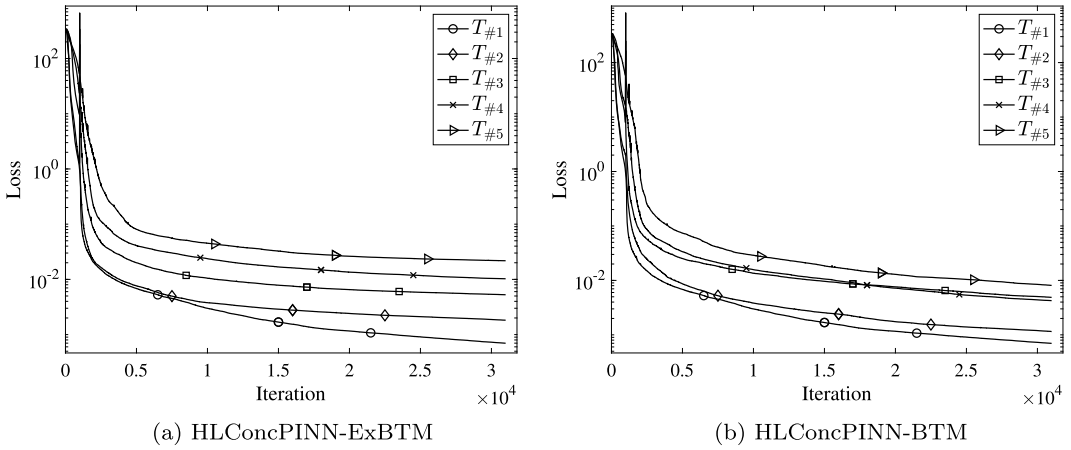


Fig. 5. Heat equation: Training loss versus the training iterations for different time blocks with the (a) HLConcPINN-ExBTM and (b) HLConcPINN-BTM methods. NN architecture: [2, 90, 90, 10, 1], tanh activation function; $N_c = 2000$ for the training collocation points. The legend shows the time block index, with e.g. $T_{\#2}$ denoting the second time block.

Table 2
Heat equation: l^2 and l^∞ errors in different time blocks corresponding to a range of training collocation points N_c for the HLConcPINN-ExBTM and HLConcPINN-BTM methods. NN architecture: $[2, 90, 90, 1]$, with tanh activation function.

Error	Time block	$N_c = 1500$		$N_c = 2000$		$N_c = 2500$		$N_c = 3000$	
		ExBTM	BTM	ExBTM	BTM	ExBTM	BTM	ExBTM	BTM
l^2	$T_{\#1}$	3.63e-04	3.63e-04	4.16e-04	4.16e-04	2.58e-04	2.58e-04	3.88e-04	3.88e-04
	$T_{\#2}$	1.00e-03	5.93e-04	5.73e-04	5.93e-04	9.14e-04	7.85e-04	4.42e-04	5.51e-04
	$T_{\#3}$	7.29e-04	6.37e-04	7.75e-04	2.93e-04	9.89e-04	1.00e-03	7.03e-04	4.86e-04
	$T_{\#4}$	5.21e-04	7.84e-04	6.93e-04	5.52e-04	8.38e-04	6.03e-04	7.49e-04	9.14e-04
	$T_{\#5}$	9.14e-04	1.03e-03	1.77e-03	1.40e-03	6.04e-04	6.01e-04	1.21e-03	1.21e-03
l^∞	$T_{\#1}$	1.64e-03	1.64e-03	2.49e-03	2.49e-03	1.31e-03	1.31e-03	2.34e-03	2.34e-03
	$T_{\#2}$	5.74e-03	5.22e-03	3.02e-03	3.43e-03	6.72e-03	5.04e-03	2.09e-03	4.82e-03
	$T_{\#3}$	4.77e-03	3.67e-03	4.16e-03	2.27e-03	4.39e-03	4.75e-03	8.61e-03	2.02e-03
	$T_{\#4}$	3.09e-03	1.45e-02	3.81e-03	5.86e-03	9.06e-03	3.84e-03	4.23e-03	5.23e-03
	$T_{\#5}$	4.70e-03	1.08e-02	3.22e-02	2.29e-02	5.23e-03	1.90e-03	6.52e-03	1.86e-02

$$
\begin{aligned}
Loss_i^{II} = & \frac{W_1}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - \Delta u_{\theta_i}(x_{int}^n, t_{int}^n) - f(x_{int}^n, t_{int}^n)\right]^2 \\
& + \frac{W_2}{N_c}\sum_{n=1}^{N_c}\left[u_{\theta_i}(x_{tb}^n, t_{i-1}) - u_{\theta_{i-1}}(x_{tb}^n, t_{i-1})\right]^2 \\
& + W_3\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}\left[(u_{\theta_i}(0, t_{sb}^n) - g_1(t_{sb}^n))^2 + (u_{\theta_i}(1, t_{sb}^n) - g_2(t_{sb}^n))^2\right]\Big)^{1/2},
\end{aligned}
\tag{67}
$$

where $u_{\theta_0}(x, t_0) = h(x)$. In subsequent simulations the penalty coefficients are fixed to $(W_1, W_2, W_3) = (0.8, 0.9, 0.9)$ in both $Loss_i^I$ and $Loss_i^{II}$, and 5 uniform time blocks are employed in block time marching. The HLConcPINN schemes employing these two distinctive loss functions will be designated as HLConcPINN-ExBTM ($Loss_i^I$) and HLConcPINN-BTM ($Loss_i^{II}$), respectively.

An overview of the solution field and the training histories is provided by Figs. 3, 4, and 5 for the HLConcPINN-ExBTM and the HLConcPINN-BTM methods. Fig. 3 shows distributions in the space-time domain of the true solution, the HLConcPINN-ExBTM solution, and the HLConcPINN-BTM solution, as well as the point-wise absolute errors of the HLConcPINN-ExBTM and HLConcPINN-BTM solutions. Fig. 4 compares profiles of the true solution, the HLConcPINN-ExBTM and HLConcPINN-BTM solutions at three time instants ($t = 2.5$, 5 and 9.5), and also show the error profiles of the HLConcPINN-ExBTM and HLConcPINN-BTM methods. Fig. 5 depicts the training loss histories for each of the 5 time blocks with the HLConcPINN-ExBTM and HLConcPINN-BTM methods. In this set of simulations, three hidden layers and the tanh activation function are employed in the neural network. The specific parameter values are provided in the captions of these figures; see also Table 1. The HLConcPINN-ExBTM and the HLConcPINN-BTM methods are able to capture the solution quite accurately, with the HLConcPINN-BTM solution appearing slightly better.

Table 2 shows a study of the effect of training collocation points on the results of the HLConcPINN-ExBTM and HLConcPINN-BTM methods. The l^2 and l^∞ errors of these methods in different time blocks obtained with collocation points ranging from $N_c = 1500$ to $N_c = 3000$ are listed in the table. Here the neural network has an architecture $[2, 90, 90, 1]$, with the tanh activation function for all

Table 3
Heat equation: l^2 and l^∞ errors in different time blocks corresponding to a series of network architectures with varying number of hidden layers for the HLConcPINN-ExBTM and HLConcPINN-BTM methods. tanh activation function; $N_c = 2000$ for the collocation points. NN architectural vectors are specified in row one of the table.

Error	Time block	[2,90,90,1]		[2,90,90,10,1]		[2,90,90,10,10,1]		[2,90,90,10,10,10,1]	
		ExBTM	BTM	ExBTM	BTM	ExBTM	BTM	ExBTM	BTM
l^2	$T_{\#1}$	4.16e-04	4.16e-04	1.79e-04	1.79e-04	1.99e-04	1.99e-04	2.44e-04	2.44e-04
	$T_{\#2}$	5.73e-04	5.93e-04	2.40e-04	3.46e-04	3.13e-04	2.03e-04	3.95e-04	3.02e-04
	$T_{\#3}$	7.75e-04	2.93e-04	5.33e-04	7.24e-04	8.17e-04	7.28e-04	8.91e-04	9.68e-04
	$T_{\#4}$	6.93e-04	5.52e-04	5.92e-04	6.30e-04	1.70e-03	7.86e-04	1.01e-03	1.44e-03
	$T_{\#5}$	1.77e-03	1.40e-03	9.06e-04	8.46e-04	1.49e-03	9.15e-04	1.59e-03	2.29e-03
l^∞	$T_{\#1}$	2.49e-03	2.49e-03	2.97e-03	2.97e-03	9.11e-04	9.11e-04	1.47e-03	1.47e-03
	$T_{\#2}$	3.02e-03	3.43e-03	2.62e-03	3.05e-03	1.43e-03	1.82e-03	1.89e-03	1.92e-03
	$T_{\#3}$	4.16e-03	2.27e-03	3.90e-03	3.94e-03	4.05e-03	4.08e-03	4.79e-03	4.92e-03
	$T_{\#4}$	3.81e-03	5.86e-03	4.24e-03	4.45e-03	1.29e-02	8.33e-03	1.04e-02	2.49e-02
	$T_{\#5}$	3.22e-02	2.29e-02	1.62e-02	4.70e-03	8.92e-03	7.20e-03	1.29e-02	1.62e-02

Table 4
Heat equation: l^2 and l^∞ errors in different time blocks of the HLConcPINN-ExBTM and HLConcPINN-BTM methods obtained with several activation functions. NN architecture: $[2, 90, 90, 10, 10, 1]$; $N_c = 2000$ for training collocation points. The activation function is fixed to tanh in the first two hidden layers, and is varied among sine, Gaussian, swish, and softplus in the subsequent hidden layers.

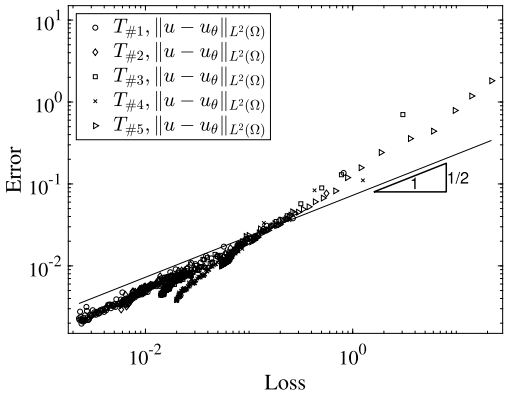
Error	Time block	Sine		Gaussian		Swish		Softplus	
		ExBTM	BTM	ExBTM	BTM	ExBTM	BTM	ExBTM	BTM
l^2	$T_{\#1}$	1.34e-04	1.34e-04	1.03e-04	1.03e-04	2.61e-04	2.61e-04	2.96e-04	2.96e-04
	$T_{\#2}$	1.53e-04	1.91e-04	1.86e-04	2.28e-04	2.56e-04	2.38e-04	3.68e-04	3.48e-04
	$T_{\#3}$	3.06e-04	2.96e-04	4.41e-04	4.11e-04	5.27e-04	4.09e-04	3.28e-04	3.90e-04
	$T_{\#4}$	6.03e-04	4.98e-04	5.80e-04	8.05e-04	7.75e-04	6.49e-04	8.40e-04	1.02e-03
	$T_{\#5}$	6.94e-04	6.30e-04	7.35e-04	8.98e-04	7.45e-04	3.22e-03	1.50e-03	1.13e-03
l^∞	$T_{\#1}$	8.18e-04	8.18e-04	1.02e-03	1.02e-03	1.44e-03	1.44e-03	1.73e-03	1.73e-03
	$T_{\#2}$	8.95e-04	1.29e-03	1.53e-03	1.52e-03	1.73e-03	1.17e-03	2.08e-03	1.33e-03
	$T_{\#3}$	8.82e-04	2.39e-03	4.22e-03	2.65e-03	3.41e-03	1.97e-03	2.86e-03	3.96e-03
	$T_{\#4}$	3.97e-03	3.06e-03	4.68e-03	3.86e-03	4.47e-03	3.35e-03	3.87e-03	4.19e-03
	$T_{\#5}$	4.03e-03	4.53e-03	2.73e-03	5.57e-03	7.40e-03	1.90e-02	6.22e-03	5.66e-03

hidden layers. The data indicate that the errors of these methods are not sensitive to the number of training collocation points. In most of subsequent tests we employ a fixed $N_c = 2000$ for the training collocation points.

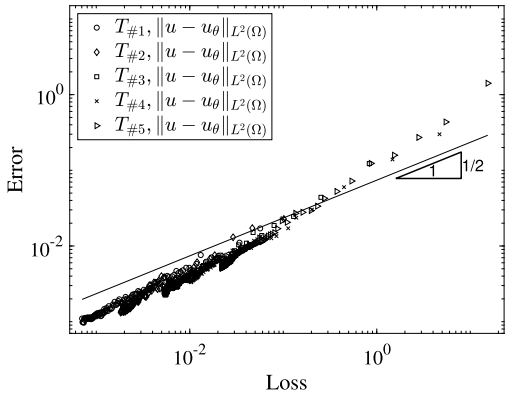
A salient feature of the current method lies in that the theoretical analyses are applicable to neural network architectures with more than two hidden layers. Table 3 shows a test of the network depth (number of hidden layers) on the HLConcPINN-ExBTM and HLConcPINN-BTM results for the heat equation. It lists the l^2 and l^∞ errors in different time blocks obtained by these methods using network architectures with 2 to 5 hidden layers. The network architectural vectors are given in the table. We employ the tanh activation function for all hidden layers, and a fixed $N_c = 2000$ for the training collocation points in these tests. We can make several observations. First, the errors grow over time with both methods. For example, the l^2 errors increase from around 10^{-4} in time block #1 to around 10^{-3} in time block #5. Second, increasing the number of hidden layers only slightly influences the accuracy of results. The errors in general appear to decrease from two to three hidden layers. As the number of hidden layers further increases to three to five, the errors tend to increase slightly compared with those of two hidden layers. Third, the errors obtained with HLConcPINN-ExBTM and HLConcPINN-BTM are generally comparable, with one slightly better than the other in different cases.

The current HLConcPINN-ExBTM method admits theoretical analyses in cases where more general activation functions are employed. Table 4 provides a study of the effect of the activation functions on the simulation results of the HLConcPINN-ExBTM and HLConcPINN-BTM methods. We employ a neural network architecture $[2, 90, 90, 10, 10, 1]$, and $N_c = 2000$ for the training collocation points. The activation function in the first two hidden layers is fixed to tanh. For the subsequent hidden layers we vary the activation function among the sine, Gaussian, swish, or softplus functions. The l^2 and l^∞ errors of HLConcPINN-ExBTM and HLConcPINN-BTM in different time blocks corresponding to these activation functions are provided in the table. These results can be compared with those in Table 3 for the same network architecture, where the tanh activation function has been used for all hidden layers. Overall, the sine activation function appears to produce the best results for HLConcPINN-ExBTM and HLConcPINN-BTM. The results obtained with the Gaussian, tanh, swish and softplus functions seem comparable to one another in terms of the accuracy.

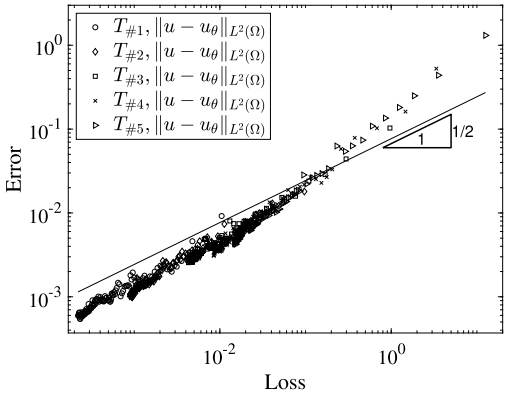
Theorem 3.5 indicates that the approximation error of the solution to the heat equation obtained with the HLConcPINN-ExBTM method scales as the square root of the training loss for all time blocks. Fig. 6 provides numerical evidence corroborating this statement. Here we plot the l^2 errors of the solution as a function of the training loss value (in logarithmic scale) for HLConcPINN-ExBTM from our simulations. The number of hidden layers varies from two to four in these tests, with tanh as the activation functions for the first two hidden layers and Gaussian or softplus for the subsequent hidden layers. We have used $N_c = 2000$ for the collocation



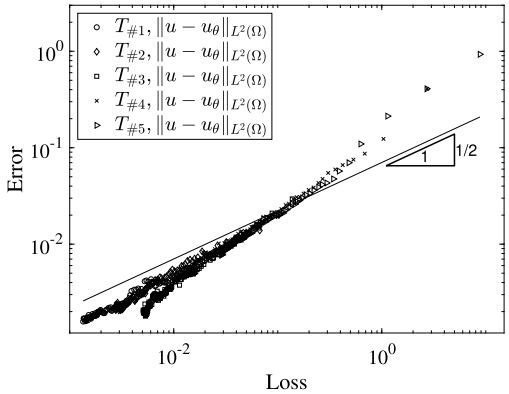
(a) NN: [2,90,90,1]. Activation: tanh in all hidden layers.



(b) NN: [2,90,90,10,1]. Activation: tanh in all hidden layers.

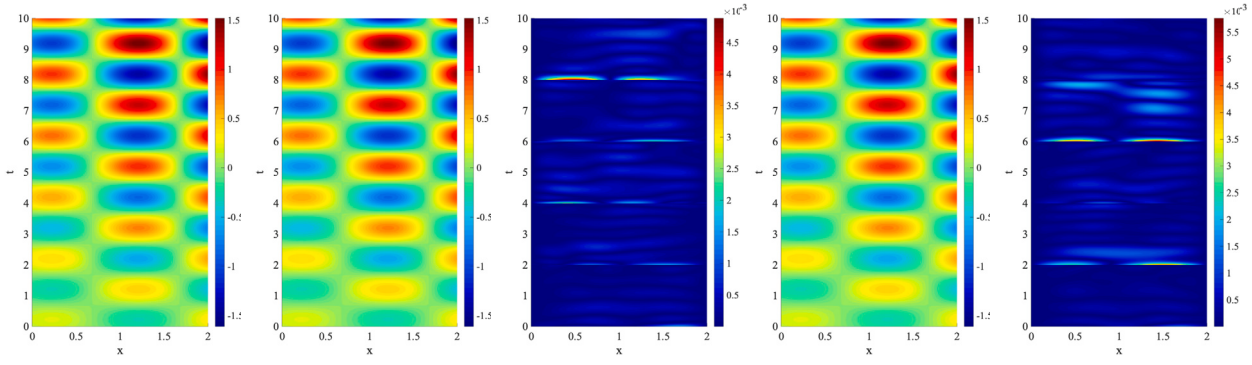


(c) NN: [2,90,90,10,10,1]. Activation: tanh in first two and Gaussian in subsequent hidden layers.



(d) NN: [2,90,90,10,10,1]. Activation: tanh in first two and softplus in subsequent hidden layers.

Fig. 6. Heat equation: l^2 errors of HLConcPINN-ExBTM as a function of the training loss values, obtained with different network architectures and activation functions. $N_c = 2000$ for the training collocation points.



(a) u (b) u_θ (c) $|u-u_\theta|$ (d) u^*_θ (e) $|u-u^*_\theta|$

Fig. 7. Burgers' equation: Distributions of the exact solution (a), the HLConcPINN-ExBTM solution and its point-wise error (b,c), and the HLConcPINN-BTM solution and its point-wise error (d,e). NN: [2,90,90,10,1], with tanh, tanh and sine activation functions for the three hidden layers; $N_c = 2500$ for the training collocation points.

points. The data generally signify a square root scaling consistent with the theoretical analysis, with some deviation (faster than square root) toward larger training loss values.

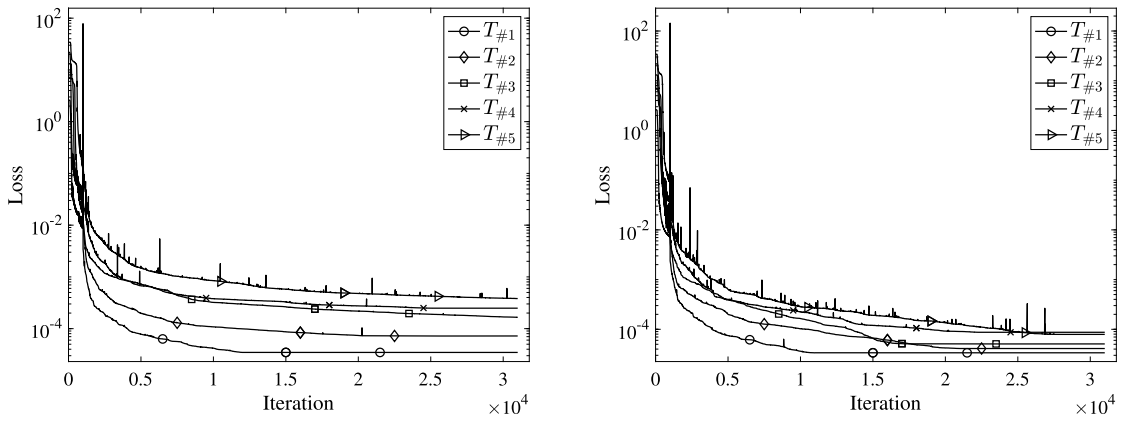


Fig. 8. Burgers' equation: Loss histories of HLConcPINN-ExBTM and HLConcPINN-BTM in different time blocks. NN settings and simulation parameters follow those of Fig. 7.

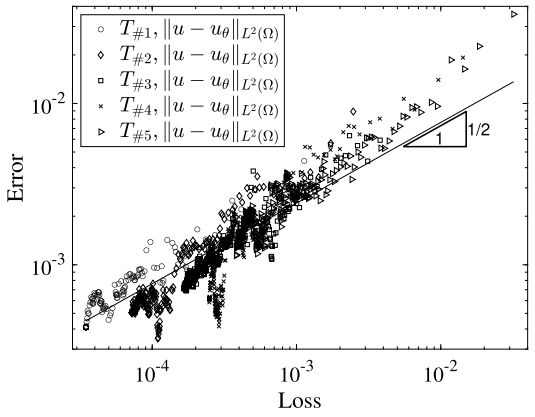


Fig. 9. Burgers' equation: The l^2 errors of u as a function of the training loss for the HLConcPINN-ExBTM method. The NN settings and simulation parameters follow those of Fig. 7.

7.2. Burgers' equation

We next consider the viscous Burgers' equation on the spatial-temporal domain $(x,t) \in \Omega = D \times [0,T] = [0,2] \times [0,10]$,

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} + u \frac{\partial u}{\partial x} = f(x,t), \tag{68a}$$

$$u(0,t) = \phi_1(t), \quad u(2,t) = \phi_2(t), \quad u(x,0) = \psi(x). \tag{68b}$$

Here $u(x,t)$ is the field to be solved for, ν denotes the viscosity, $f(x,t)$ is a source term, $\phi_1(t)$ and $\phi_2(t)$ denote the boundary data, and $\psi(x)$ is the initial distribution. We take $\nu = 1$ and choose the source term and the boundary/initial condition such that the function

$$u(x,t) = \left(\frac{1}{5} + \frac{x}{10}\right)\left(\frac{1}{5} + \frac{t}{10}\right)\left[2\sin\left(\pi x + \frac{2\pi}{5}\right) + \frac{1}{2}\cos\left(\pi x - \frac{3\pi}{5}\right)\right]\left[2\sin\left(\pi t + \frac{2\pi}{5}\right) + \frac{1}{2}\cos\left(\pi t - \frac{3\pi}{5}\right)\right],$$

solves the problem (68).

The loss function for the HLConcPINN-ExBTM method is given by,

$$\begin{aligned} Loss_i^I =& \frac{W_1}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - \nu\frac{\partial^2 u_{\theta_i}}{\partial x^2}(x_{int}^n, t_{int}^n) + u_{\theta_i}(x_{int}^n, t_{int}^n)\frac{\partial u_{\theta_i}}{\partial x}(x_{int}^n, t_{int}^n) - f(x_{int}^n, t_{int}^n)\right]^2 \\ &+ \frac{W_2}{N_c}\sum_{j=1}^{i}\sum_{n=1}^{N_c}\left[u_{\theta_i}(x_{tb}^n, t_{j-1}) - u_{\theta_{j-1}}(x_{tb}^n, t_{j-1})\right]^2 \\ &+ \frac{W_3}{N_c}\sum_{n=1}^{N_c}\left[(u_{\theta_i}(0, t_{sb}^n) - g_1(t_{sb}^n))^2 + (u_{\theta_i}(2, t_{sb}^n) - g_2(t_{sb}^n))^2\right] \end{aligned}$$

Table 5
Burgers' equation: The l^2 and l^∞ errors corresponding to different training collocation points N_c. NN: [2,90,90,10,1], with tanh, tanh and sine activation functions for the three hidden layers.

Error	Time block	$N_c = 1500$		$N_c = 2000$		$N_c = 2500$		$N_c = 3000$	
		ExBTM	BTM	ExBTM	BTM	ExBTM	BTM	ExBTM	BTM
l^2	$T_{\#1}$	2.26e-03	2.41e-03	1.19e-03	1.29e-03	1.14e-03	1.08e-03	1.75e-03	1.62e-03
	$T_{\#2}$	6.31e-04	1.72e-03	6.92e-04	6.40e-04	8.35e-04	1.80e-03	6.50e-04	4.54e-04
	$T_{\#3}$	7.05e-04	7.63e-04	7.72e-04	7.59e-04	8.32e-04	6.74e-04	8.69e-04	9.56e-04
	$T_{\#4}$	1.60e-03	8.04e-04	6.76e-04	6.78e-04	5.61e-04	1.48e-03	7.45e-04	8.76e-04
	$T_{\#5}$	1.74e-03	1.81e-03	4.77e-04	1.32e-03	8.50e-04	4.81e-04	8.23e-04	1.17e-03
l^∞	$T_{\#1}$	2.01e-02	1.97e-02	1.03e-02	1.17e-02	8.77e-03	7.49e-03	1.23e-02	4.28e-03
	$T_{\#2}$	4.43e-03	9.01e-03	3.68e-03	2.84e-03	5.71e-03	1.46e-02	4.61e-03	3.63e-03
	$T_{\#3}$	4.58e-03	6.60e-03	7.41e-03	4.13e-03	6.32e-03	3.58e-03	5.27e-03	7.47e-03
	$T_{\#4}$	5.06e-03	3.36e-03	6.37e-03	4.61e-03	4.52e-03	1.05e-02	4.13e-03	6.44e-03
	$T_{\#5}$	1.28e-02	1.09e-02	2.36e-03	3.40e-03	7.24e-03	1.52e-03	3.64e-03	3.31e-03

Table 6
Burgers' equation: The l^2 and l^∞ errors obtained with several different activation functions. NN: [2,90,90,10,1], with tanh activation function for the first two hidden layers, while the activation function for the last hidden layer is varied as given in the table. $N_c = 2500$ training collocation points.

Error	Time block	tanh		Gaussian		Swish		Softplus	
		ExBTM	BTM	ExBTM	BTM	ExBTM	BTM	ExBTM	BTM
l^2	$T_{\#1}$	1.44e-03	1.04e-03	1.59e-03	2.23e-03	1.60e-03	2.23e-03	2.10e-03	1.80e-03
	$T_{\#2}$	9.66e-04	1.19e-03	1.01e-03	2.15e-03	2.35e-03	3.96e-03	8.69e-04	7.42e-04
	$T_{\#3}$	1.69e-03	8.65e-04	1.39e-03	5.70e-04	1.87e-03	1.12e-03	1.77e-03	1.37e-03
	$T_{\#4}$	1.05e-03	1.26e-03	1.05e-03	1.07e-03	1.42e-03	1.22e-03	2.31e-03	1.04e-03
	$T_{\#5}$	1.66e-03	2.13e-03	1.32e-03	2.89e-03	2.55e-03	1.53e-03	2.41e-03	1.39e-03
l^∞	$T_{\#1}$	9.19e-03	6.56e-03	1.69e-02	2.26e-02	1.02e-02	2.00e-02	1.87e-02	1.31e-02
	$T_{\#2}$	8.05e-03	1.19e-02	4.97e-03	1.83e-02	3.22e-02	4.87e-02	4.43e-03	2.78e-03
	$T_{\#3}$	2.23e-02	7.64e-03	1.68e-02	5.14e-03	2.20e-02	1.11e-02	9.10e-03	1.34e-02
	$T_{\#4}$	5.90e-03	1.11e-02	9.88e-03	6.82e-03	1.58e-02	6.79e-03	2.00e-02	6.14e-03
	$T_{\#5}$	1.33e-02	1.82e-02	1.14e-02	9.61e-03	1.47e-02	5.31e-03	9.31e-03	4.58e-03

$$+ W_4\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}(u_{\theta_i}(0,t_{sb}^n) - g_1(t_{sb}^n))^2\Big)^{1/2} + W_5\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}(u_{\theta_i}(2,t_{sb}^n) - g_2(t_{sb}^n))^2\Big)^{1/2} + Loss_{i-1}^I, \tag{69}$$

where $Loss_0^I = 0$, and we have added a set of penalty coefficients $W_k > 0$ $(k = 1, \cdots, 5)$ for different loss terms. The loss function for HLConcPINN-BTM is,

$$\begin{aligned} Loss_i^{II} =& \frac{W_1}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - \nu\frac{\partial^2 u_{\theta_i}}{\partial x^2}(x_{int}^n, t_{int}^n) + u_{\theta_i}(x_{int}^n, t_{int}^n)\frac{\partial u_{\theta_i}}{\partial x}(x_{int}^n, t_{int}^n) - f(x_{int}^n, t_{int}^n)\right]^2 \\ &+ \frac{W_2}{N_c}\sum_{n=1}^{N_c}\left[u_{\theta_i}(x_{tb}^n, t_{i-1}) - u_{\theta_{j-1}}(x_{tb}^n, t_{i-1})\right]^2 \\ &+ \frac{W_3}{N_c}\sum_{n=1}^{N_c}\left[(u_{\theta_i}(0,t_{sb}^n) - g_1(t_{sb}^n))^2 + (u_{\theta_i}(2,t_{sb}^n) - g_2(t_{sb}^n))^2\right] \\ &+ W_4\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}(u_{\theta_i}(0,t_{sb}^n) - g_1(t_{sb}^n))^2\Big)^{1/2} + W_5\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}(u_{\theta_i}(2,t_{sb}^n) - g_2(t_{sb}^n))^2\Big)^{1/2}. \end{aligned} \tag{70}$$

For both methods, we employ $(W_1, \cdots, W_5) = (0.6, 0.4, 0.4, 0.4, 0.4)$ in the following simulations. Five uniform time blocks are employed in block time marching.

Fig. 7 shows distributions of the true solution, the HLConcPINN-ExBTM and HLConcPINN-BTM solutions and their absolute errors. The neural network structure and other parameters are provided in the figure caption. The histories of the training loss functions for HLConcPINN-ExBTM and HLConcPINN-BTM are shown in Fig. 8. Both methods have captured the solution well.

Tables 5 and 6 illustrate the effects of the training collocation points and the activation function on the simulation points. In these simulations the neural network structure is characterized by $[2, 90, 90, 10, 1]$, with the activation function tanh for the first two hidden layers. In Table 5, the activation function for the last hidden layer is set to sine, and the number of training collocation points is varied systematically. In Table 6 the activation function for the last hidden layer is varied (tanh, Gaussian, swish, or softplus), with

Table 7
Burgers' equation: The l^2 and l^∞ errors and the running time with HLConcPINN and FEM. NN architecture in HLConcPINN: $[2, 90, 90, 10, 1]$; $N_c = 2000$ for training collocation points; the activation function is fixed to tanh in the first two hidden layers and sine in the third hidden layer. Temporal and spatial settings in FEM: both the temporal and spatial directions are uniformly divided into 90 segments in each time block. "ExBTM": HLConcPINN-ExBTM; "BTM": HLConcPINN-BTM.

	l^2			l^∞			Wall time (seconds)		
	ExBTM	BTM	FEM	ExBTM	BTM	FEM	ExBTM	BTM	FEM
$T_{\#1}$	1.19e-03	1.29e-03	7.05e-04	1.03e-02	1.17e-02	1.34e-03	2069.0055	2043.5774	0.1673
$T_{\#2}$	6.92e-04	6.40e-04	6.65e-04	3.68e-03	2.84e-03	1.33e-03	2178.0739	2070.9914	0.1688
$T_{\#3}$	7.72e-04	7.59e-04	6.64e-04	7.41e-03	4.13e-03	1.39e-03	2702.4028	2032.0701	0.1604
$T_{\#4}$	6.76e-04	6.78e-04	6.72e-04	6.37e-03	4.61e-03	1.46e-03	2792.7288	2036.1165	0.1651
$T_{\#5}$	4.77e-04	1.32e-03	6.85e-04	2.36e-03	3.40e-03	1.53e-03	3007.0272	2029.8013	0.1603

fixed training collocation points $N_c = 2500$. The simulation results appear not sensitive to the training collocation points, similar to observations with the previous test problem. Among the activation functions tested, the sine function appears to produce the best result.

Fig. 9 illustrates the relation between the l^2 error of u and the training loss value for different time blocks obtained with the HLConcPINN-ExBTM method in our simulations. The scaling manifested in the data is consistent with Theorem 4.3 from our analyses.

The network training in the current HLConcPINN methods, similar to other PINN type methods, requires a large number of iterations using the Adam or L-BFGS optimizer. As a result, the training cost would be significantly higher than that using traditional numerical methods such as the finite element method (FEM). In Table 7 we show a comparison of the l^2 and l^∞ errors between the current HLConcPINN-ExBTM and HLConcPINN-BTM methods and the classical FEM (2nd-order), as well as the computational cost (network training time of current algorithms, and FEM computation time), for solving the Burgers' equation. For the current methods, we employ five uniform time blocks in block time marching. Within each time block $N_c = 2000$ random collocation points are used in the interior of the space-time domain and on each of the domain boundaries (2 spatial boundaries and 1 boundary in time), leading to a total of 8000 collocation points. We employ a neural network architecture $[2, 90, 90, 10, 1]$, with the tanh activation function for the first two hidden layers and the sine activation function for the last hidden layer. Our FEM solver is implemented in FEniCSx [2]. For FEM, we divide the temporal dimension into five uniform blocks and the computation is carried out block by block, in a way analogous to block time marching. On each time block, the Burgers' equation is discretized using $\mathbb{P}_1$ Lagrange elements in space and the second-order backward differentiation formula (BDF2) in time with a semi-implicit strategy, in which the nonlinear term is treated explicitly and the viscous term is treated implicitly. We employ 90 elements in space and 90 time steps within each time block. This amounts to approximately $90 \times 91 = 8190$ nodal points in FEM, which is comparable to the total number of collocation points employed in the current HLConcPINN methods for each time block. The l^2 and l^∞ errors and the wall time for different time blocks are listed in the table. The error levels produced by the current methods and the FEM are largely comparable, with the FEM a little better for a number of cases (especially with the l^∞ error). On the other hand, the network training cost of the current HLConcPINN methods is much larger than the computation time of FEM.

7.3. Wave equation

We next simulate the wave equation in one spatial dimension (plus time) using the current method, following a configuration from [15]. Consider the spatial-temporal domain, $(x,t) \in D \times [0,T] = [0,5] \times [0,10]$, and the following initial-boundary value problem on this domain,

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0, \tag{71a}$$

$$u(0,t) = u(5,t), \quad \frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(5,t), \quad u(x,0) = 2\,\mathrm{sech}^3\left(\frac{3}{\delta_0}(x - x_0)\right), \quad \frac{\partial u}{\partial t}(x,0) = 0, \tag{71b}$$

where $u(x,t)$ is the wave field to be solved for, c is the wave speed, x_0 is the initial peak location of the wave, δ_0 is a constant that controls the width of the wave profile, and periodic boundary conditions are imposed on $x = 0$ and 5. We employ $c = 2$, $\delta_0 = 2$, and $x_0 = 3$ for this problem. This problem has the following solution

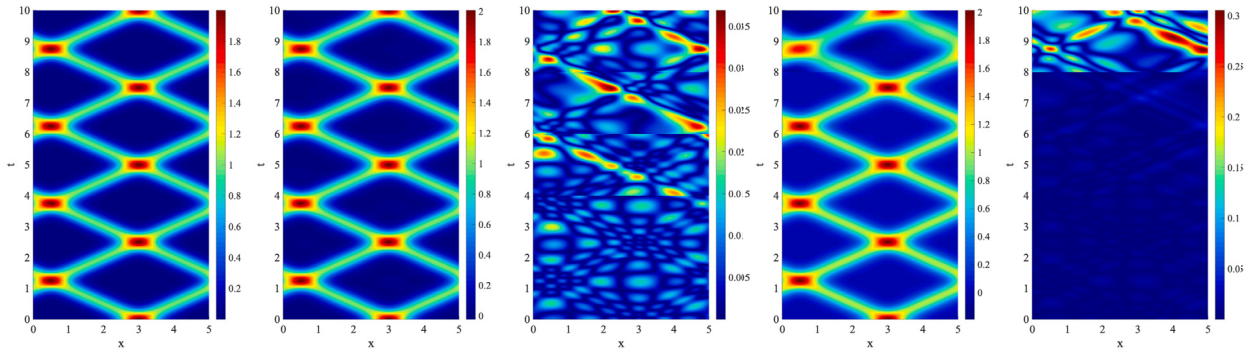
$$\begin{cases} u(x,t) = \mathrm{sech}^3\left(\frac{3}{\delta_0}(-2.5 + \xi)\right) + \mathrm{sech}^3\left(\frac{3}{\delta_0}(-2.5 + \eta)\right), \\ \xi = \mathrm{mod}\left(x - x_0 + ct + 2.5, 5\right), \quad \eta = \mathrm{mod}\left(x - x_0 - ct + 2.5, 5\right), \end{cases}$$

where mod refers to the modulo operation. In the simulations we introduce the auxiliary field $v(x,t)$ and rewrite (71) into

$$\frac{\partial u}{\partial t} - v = 0, \qquad \frac{\partial v}{\partial t} - c^2 \frac{\partial^2 u}{\partial x^2} = 0, \tag{72a}$$

$$u(0,t) = u(5,t), \quad \frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(5,t), \quad u(x,0) = 2\,\mathrm{sech}^3\left(\frac{3}{\delta_0}(x - x_0)\right), \quad v(x,0) = 0, \tag{72b}$$

where $v(x,t)$ is defined by the first equation in (72a).



(a) u (b) u_θ (c) $|u-u_\theta|$ (d) u_θ^* (e) $|u-u_\theta^*|$

Fig. 10. Wave equation: Solution distributions (u: true solution; u_θ: HLConcPINN-ExBTM solution; u_θ^* HLConcPINN-BTM solution). NN: $[2, 90, 90, 10, 2]$; activation function: tanh for the first two hidden layers, sine for the last hidden layer; $N_c = 2500$ for training collocation points.

To simulate the system (72), the training error in (40)– (41) leads to the following loss function with the HLConcPINN-ExBTM method for the i-th time block ($1 \le i \le I$),

$$
\begin{aligned}
Loss_i^I =& \frac{W_1}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - v_{\theta_i}(x_{int}^n, t_{int}^n)\right]^2 + \frac{W_2}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial v_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - 4\frac{\partial^2 u_{\theta_i}}{\partial x^2}(x_{int}^n, t_{int}^n)\right]^2 \\
&+ \frac{W_3}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial^2 u_{\theta_i}}{\partial t \partial x}(x_{int}^n, t_{int}^n) - \frac{\partial v_{\theta_i}}{\partial x}(x_{int}^n, t_{int}^n)\right]^2 + \frac{W_4}{N_c}\sum_{j=1}^{i}\sum_{n=1}^{N_c}\left[u_{\theta_i}(x_{tb}^n, t_{j-1}) - u_{\theta_{j-1}}(x_{tb}^n, t_{j-1})\right]^2 \\
&+ \frac{W_5}{N_c}\sum_{j=1}^{i}\sum_{n=1}^{N_c}\left[v_{\theta_i}(x_{tb}^n, t_{j-1}) - v_{\theta_{j-1}}(x_{tb}^n, t_{j-1})\right]^2 + \frac{W_6}{N_c}\sum_{j=1}^{i}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial x}(x_{tb}^n, t_{j-1}) - \frac{\partial u_{\theta_{j-1}}}{\partial x}(x_{tb}^n, t_{j-1})\right]^2 \\
&+ W_7\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}\left[v_\theta(0, t_{sb}^n) - v_\theta(5, t_{sb}^n)\right]^2\Big)^{1/2} + W_8\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial x}(0, t_{sb}^n) - \frac{\partial u_{\theta_i}}{\partial x}(5, t_{sb}^n)\right]^2\Big)^{1/2} \\
&+ Loss_{i-1}^I,
\end{aligned} \tag{73}
$$

where $Loss_0^I = 0$, and $W_k > 0$ ($1 \le k \le 8$) are the penalty coefficients added for different loss terms. The loss function with the HLConcPINN-BTM method is,

$$
\begin{aligned}
Loss_i^{II} =& \frac{W_1}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - v_{\theta_i}(x_{int}^n, t_{int}^n)\right]^2 + \frac{W_2}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial v_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - 4\frac{\partial^2 u_{\theta_i}}{\partial x^2}(x_{int}^n, t_{int}^n)\right]^2 \\
&+ \frac{W_3}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial^2 u_{\theta_i}}{\partial t \partial x}(x_{int}^n, t_{int}^n) - \frac{\partial v_{\theta_i}}{\partial x}(x_{int}^n, t_{int}^n)\right]^2 + \frac{W_4}{N_c}\sum_{n=1}^{N_c}\left[u_{\theta_i}(x_{tb}^n, t_{i-1}) - u_{\theta_{i-1}}(x_{tb}^n, t_{i-1})\right]^2 \\
&+ \frac{W_5}{N_c}\sum_{n=1}^{N_c}\left[v_{\theta_i}(x_{tb}^n, t_{i-1}) - v_{\theta_{i-1}}(x_{tb}^n, t_{i-1})\right]^2 + \frac{W_6}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial x}(x_{tb}^n, t_{i-1}) - \frac{\partial u_{\theta_{i-1}}}{\partial x}(x_{tb}^n, t_{i-1})\right]^2 \\
&+ W_7\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}\left[v_\theta(0, t_{sb}^n) - v_\theta(5, t_{sb}^n)\right]^2\Big)^{1/2} + W_8\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial x}(0, t_{sb}^n) - \frac{\partial u_{\theta_i}}{\partial x}(5, t_{sb}^n)\right]^2\Big)^{1/2}.
\end{aligned} \tag{74}
$$

In the simulations, we employ neural network architectures with two output nodes, representing the wave field u and the wave speed $v = \frac{\partial u}{\partial t}$, respectively. The penalty coefficients in the loss functions are taken to be $(W_1, ..., W_8) = (0.9, 0.9, 0.9, 0.1, 0.1, 0.1, 0.1, 0.1)$. We employ 5 uniform time blocks in block time marching. The neural network parameters (network depth/width, and activation functions) and the training collocation points are varied in the tests. The adopted neural network structures are listed in Table 1.

An overview of the HLConcPINN-ExBTM and HLConcPINN-BTM solutions to the wave equation and their accuracy is provided in Figs. 10 to 14. Figs. 10 and 11 show distributions of the wave field u and the wave speed v, corresponding to the true solution, the HLConcPINN-ExBTM and HLConcPINN-BTM solutions, as well as their point-wise absolute errors, in the spatial-temporal domain. The neural network architecture is specified in the caption of Fig. 10, consisting of three hidden layers, with the tanh activation function for the first two hidden layers and the sine function for the last hidden layer. $N_c = 2500$ has been employed for the training collocation points. The HLConcPINN-ExBTM method is observed to produce more accurate results than HLConcPINN-BTM, especially toward later time instants. The errors of both methods are observed to grow over time. In particular, the accuracy of HLConcPINN-BTM in the last time block becomes quite poor, with pronounced deviations from the true solution in the wave speed distribution.

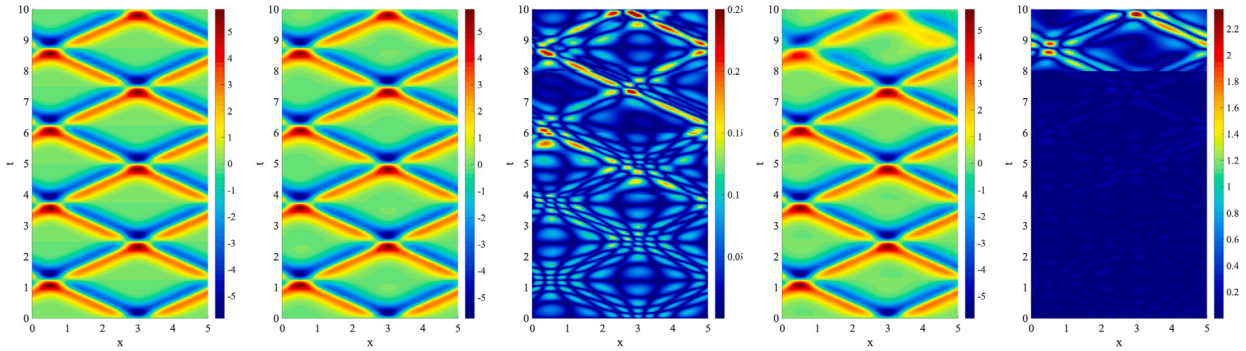


Fig. 11. Wave equation: Wave speed distributions ($v = \frac{\partial u}{\partial t}$: true solution; v_θ: HLConcPINN-ExBTM solution; v_θ^*: HLConcPINN-BTM solution). Simulation parameters follow those of Fig. 10.

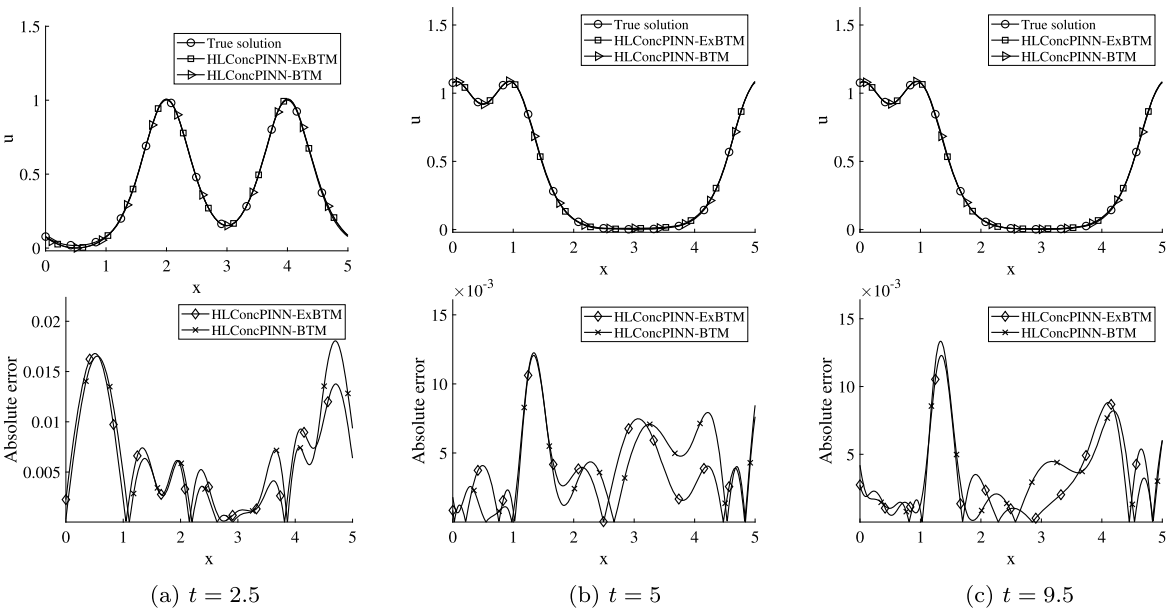


Fig. 12. Wave equation: Top row, comparison of wave profiles between the true solution and the HLConcPINN-ExBTM/-BTM solutions at several time instants. Bottom row, absolute-error profiles of HLConcPINN-ExBTM/-BTM. Simulation parameters follow those of Fig. 10.

Figs. 12 and 13 illustrate the solution profiles of the wave field u and the wave speed v obtained using HLConcPINN-ExBTM and HLConcPINN-BTM at three time instants ($t = 2.5, 5, 9.5$), accompanied by their corresponding absolute errors. The simulation parameters here follow those of Fig. 10. The error of HLConcPINN-ExBTM is generally observed to be smaller than that of HLConcPINN-BTM. The training loss histories with this group of tests for HLConcPINN-ExBTM and HLConcPINN-BTM are shown in Fig. 14. It can be generally observed that the training process results in higher loss values in later time blocks, implying a growth in the errors over time consistent with what is observed in Figs. 10 and 11.

Table 8 shows a study of the effect of the training data points on the simulation accuracy of the HLConcPINN-ExBTM and HLConcPINN-BTM methods. Here we list the l^2 and l^∞ errors of HLConcPINN-ExBTM and HLConcPINN-BTM in different time blocks obtained with training collocation points ranging from $N_c = 1500$ to $N_c = 3000$ in the simulations. The neural network architecture is given by $[2, 90, 90, 10, 2]$, with the tanh activation function for the first two hidden layers and sine function for the last hidden layer. The data suggest little sensitivity with respect to number of training data points in the range tested here.

Table 9 illustrates a test of the effect of different activation functions on the simulation results. The network architecture is characterized by $[2, 90, 90, 10, 2]$, with tanh as the activation function for the first two hidden layers, while the activation function for the last hidden layer is varied among tanh, Gaussian, swish, and softplus functions. The training collocation points are set to $N_c = 2500$. The table provides the l^2 and l^∞ errors of the wave field in different time blocks computed using HLConcPINN-ExBTM and HLConcPINN-BTM corresponding to different activation functions for the last hidden layer. These data can be compared with that of Table 8 corresponding to $N_c = 2500$, where the sine activation function has been used. Overall the sine function appears to yield the best results among the activation functions tested here.

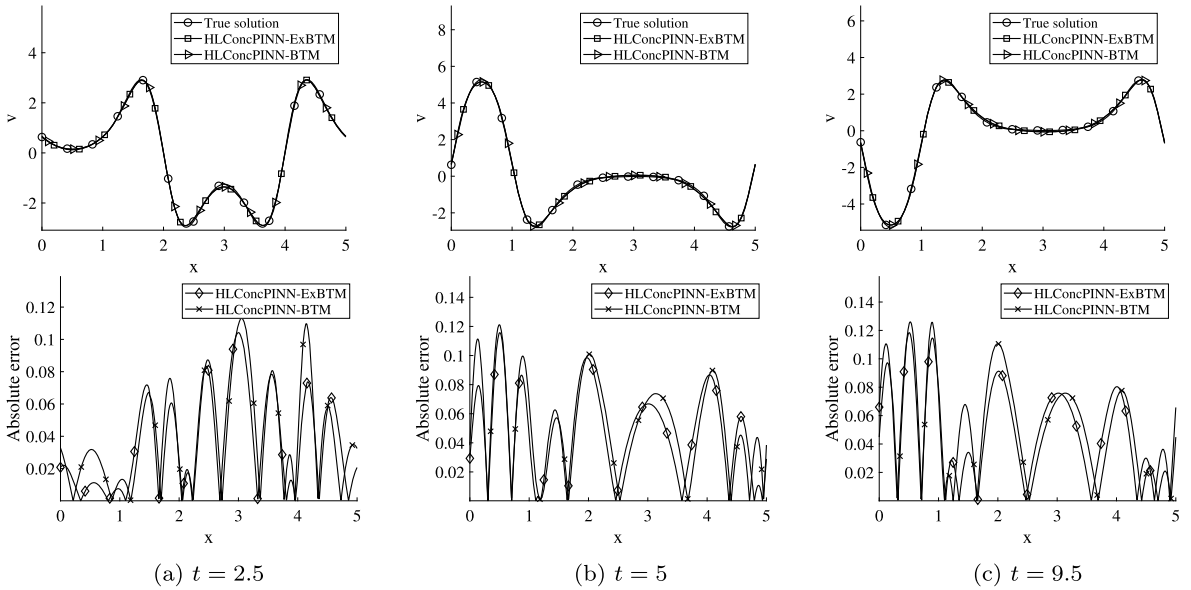


Fig. 13. Wave equation: Top row, comparison of wave speed (v) profiles between the true solution and the HLConcPINN-ExBTM/BTM solutions at several time instants. Bottom row, profiles of the absolute error of HLConcPINN-ExBTM/-BTM for v. Simulation parameters follow those of Fig. 10.

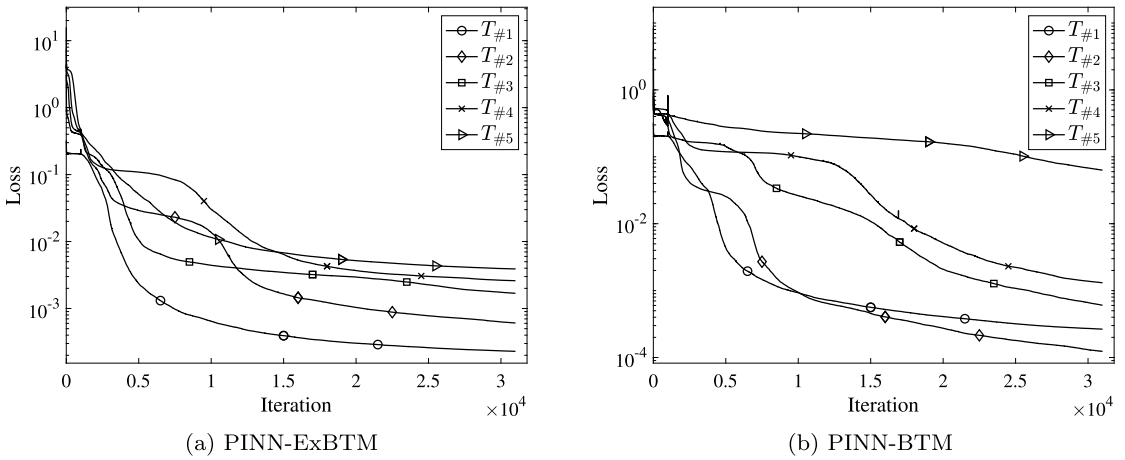


Fig. 14. Wave equation: Training loss histories in different time blocks of (a) HLConcPINN-ExBTM and (b) HLConcPINN-BTM. Simulation parameters follow those of Fig. 10.

Table 8
Wave equation: l^2 and l^∞ errors of wave field u in different time blocks obtained with HLConcPINN-ExBTM and HLConcPINN-BTM for a range of training data points N_c. NN: [2,90,90,10,2], tanh activation in first two hidden layers and sine activation in the last hidden layer.

Error	Time block	$N_c = 1500$		$N_c = 2000$		$N_c = 2500$		$N_c = 3000$	
		ExBTM	BTM	ExBTM	BTM	ExBTM	BTM	ExBTM	BTM
l^2	$T_{\#1}$	9.86e-03	1.05e-02	9.76e-03	1.01e-02	1.09e-02	1.19e-02	9.87e-03	9.31e-03
	$T_{\#2}$	1.21e-02	1.30e-02	1.14e-02	1.15e-02	1.17e-02	9.53e-03	1.01e-02	9.49e-03
	$T_{\#3}$	1.21e-02	3.73e-02	1.39e-02	1.27e-02	1.71e-02	1.52e-02	1.38e-02	1.39e-02
	$T_{\#4}$	1.75e-02	2.85e-01	5.91e-02	2.44e-02	1.74e-02	2.17e-02	5.26e-02	3.23e-01
	$T_{\#5}$	3.34e-02	3.85e-01	1.25e-01	2.15e-01	1.88e-02	1.76e-01	5.70e-02	7.36e-01
l^∞	$T_{\#1}$	3.13e-02	2.73e-02	2.86e-02	2.76e-02	2.85e-02	3.13e-02	2.75e-02	2.52e-02
	$T_{\#2}$	3.13e-02	3.57e-02	3.22e-02	3.43e-02	3.46e-02	2.64e-02	2.85e-02	2.72e-02
	$T_{\#3}$	3.43e-02	1.00e-01	6.94e-02	3.98e-02	5.47e-02	4.59e-02	4.28e-02	4.90e-02
	$T_{\#4}$	5.31e-02	8.33e-01	1.40e-01	7.40e-02	5.85e-02	6.56e-02	1.44e-01	1.13e+00
	$T_{\#5}$	8.55e-02	1.14e+00	2.28e-01	6.42e-01	6.09e-02	5.04e-01	1.65e-01	2.35e+00

Table 9
Wave equation: l^2 and l^∞ errors of the wave field u in different time blocks obtained using HLConcPINN-ExBTM and HLConcPINN-BTM with different activation functions in the last hidden layer. NN: [2,90,90,10,2], with tanh activation in the first two hidden layers. The activation function in the last hidden layer is varied, as listed in the first row of the table. $N_c = 2500$ for the training collocation points.

Error	Time block	tanh		Gaussian		Swish		Softplus	
		ExBTM	BTM	ExBTM	BTM	ExBTM	BTM	ExBTM	BTM
l^2	$T_{\#1}$	1.94e-02	2.08e-02	1.34e-02	8.44e-03	2.03e-02	1.88e-02	1.22e-02	1.51e-02
	$T_{\#2}$	2.24e-02	2.12e-02	6.07e-02	3.18e-02	2.65e-02	3.20e-02	1.66e-02	2.21e-02
	$T_{\#3}$	1.22e+00	7.27e-01	9.41e-01	4.80e-02	5.47e-02	1.65e+00	3.64e-02	3.73e-02
	$T_{\#4}$	2.84e+00	1.52e+00	2.21e+00	4.06e-01	8.17e-02	4.18e+00	9.31e-02	3.95e-01
	$T_{\#5}$	4.91e+00	2.34e+00	3.74e+00	5.42e-01	1.42e-01	7.25e+00	2.82e-01	5.83e-01
l^∞	$T_{\#1}$	5.46e-02	5.68e-02	3.74e-02	2.40e-02	5.57e-02	5.12e-02	3.54e-02	4.17e-02
	$T_{\#2}$	6.13e-02	6.77e-02	2.38e-01	8.23e-02	7.70e-02	8.74e-02	4.92e-02	6.82e-02
	$T_{\#3}$	2.66e+00	1.83e+00	2.34e+00	1.35e-01	1.27e-01	3.91e+00	1.09e-01	1.06e-01
	$T_{\#4}$	4.65e+00	2.63e+00	3.23e+00	1.41e+00	2.35e-01	6.33e+00	2.44e-01	1.43e+00
	$T_{\#5}$	7.40e+00	3.70e+00	5.16e+00	1.80e+00	4.23e-01	1.04e+01	7.86e-01	1.98e+00

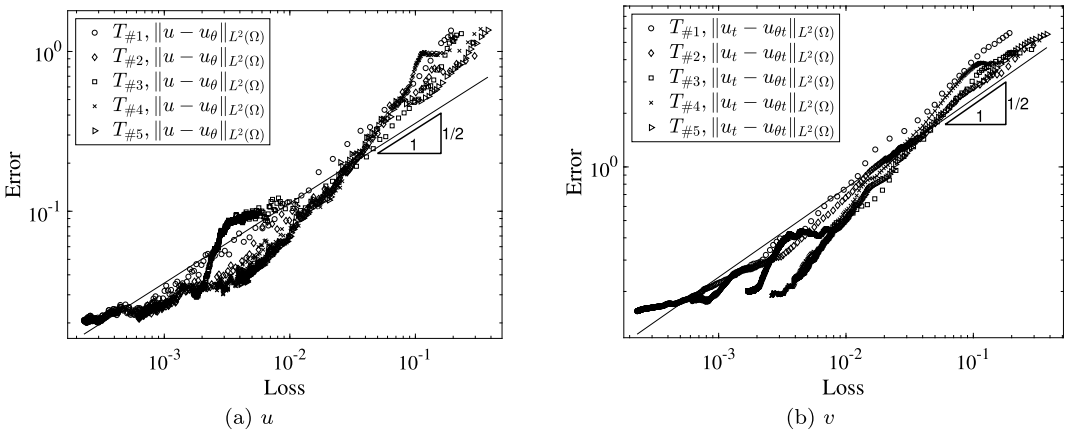


Fig. 15. Wave equation: l^2 errors of (a) the wave field u, and (b) the wave speed $v = \partial u/\partial t$, as a function of the training loss for HLConcPINN-ExBTM. NN: [2,90,90,10,2], with tanh activation for the first two hidden layers and sine activation for the last hidden layer; $N_c = 2500$ for the training data points.

Fig. 15 illustrates the relation (in logarithmic scale) between the l^2 errors of the wave field u and the wave speed $v = \frac{\partial u}{\partial t}$ as a function of the training loss value for the HLConcPINN-ExBTM method. The neural network architecture, the activation functions, and the training data points are provided in the figure caption. The simulation data approximately exhibit a scaling power of 1/2, roughly consistent with the conclusion of Theorem 5.3.

7.4. Nonlinear Klein-Gordon equation

We consider the spatial-temporal domain $(x,t) \in \Omega = D \times [0,T] = [0,1] \times [0,10]$, and the following initial/boundary value problem on this domain,

$$\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} + u + \sin(u) = f(x,t), \tag{75a}$$

$$u(0,t) = \phi_1(t), \quad u(1,t) = \phi_2(t), \quad u(x,0) = \psi_1(x), \quad \frac{\partial u}{\partial t}(x,0) = \psi_2(x). \tag{75b}$$

In these equations, $u(x,t)$ is the field function to be solved for, $f(x,t)$ is a source term, ψ_1 and ψ_2 are the initial conditions, and ϕ_1 and ϕ_2 are the boundary conditions. Note that a nonlinear term, $g(u) = \sin u$, has been used, leading to the Sine-Gordon equation in (75a). The source term, initial and boundary conditions are appropriately chosen such that the problem has the following exact solution,

$$u(x,t) = \left[2\cos\left(\pi x + \frac{\pi}{5}\right) + \frac{9}{5}\cos\left(2\pi x + \frac{7\pi}{20}\right)\right]\left[2\cos\left(\pi t + \frac{\pi}{5}\right) + \frac{9}{5}\cos\left(2\pi t + \frac{7\pi}{20}\right)\right]. \tag{76}$$

To simulate this problem, we reformulate it as follows,

$$\frac{\partial u}{\partial t} - v = 0, \quad \frac{\partial v}{\partial t} - \frac{\partial^2 u}{\partial x^2} + u + \sin(u) = f(x,t), \tag{77a}$$

$$u(0,t) = \phi_1(t), \quad u(1,t) = \phi_2(t), \quad u(x,0) = \psi_1(x), \quad v(x,0) = \psi_2(x), \tag{77b}$$

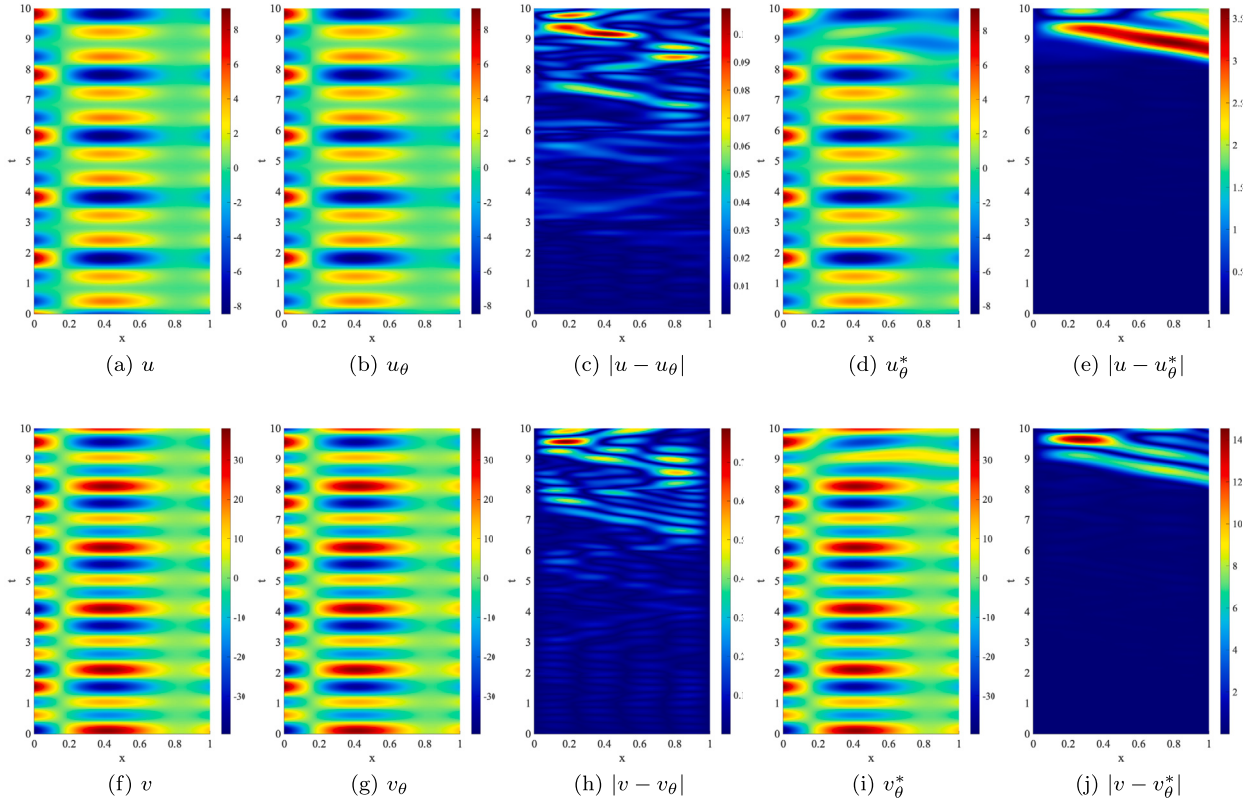


Fig. 16. Nonlinear Klein-Gordon equation: Distributions of the exact solution (a,f), the HLConcPINN-ExBTM solution and its point-wise error (b,g and c,h), and the HLConcPINN-BTM solution and its point-wise error (d,i and e,j), for u (top row) and v (bottom row). NN: [2,90,90,10,2], with tanh activation function for the first two hidden layers and sine activation function in the last hidden layer; $N_c = 2500$ for the training collocation points.

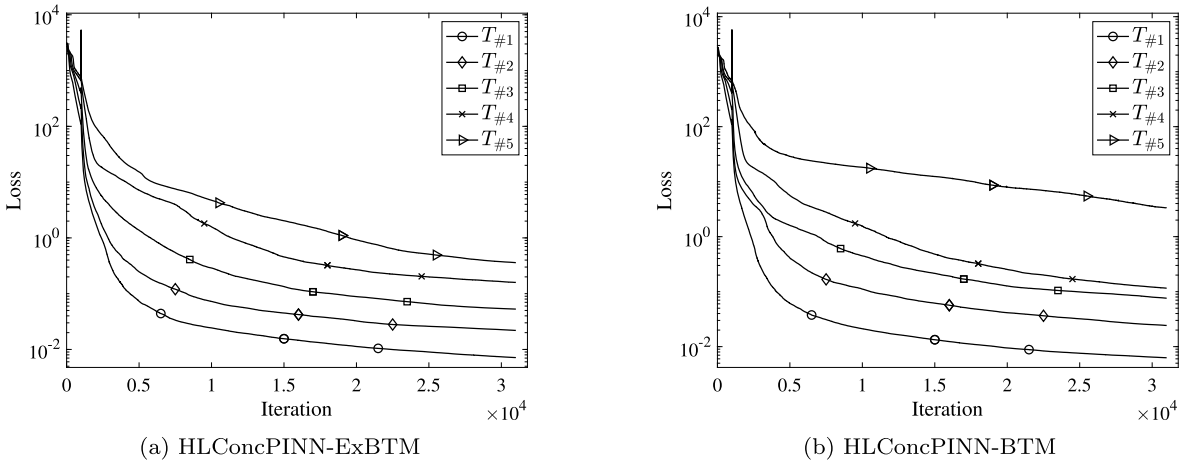


Fig. 17. Nonlinear Klein-Gordon equation: Histories of training loss for (a) HLConcPINN-ExBTM and (b) HLConcPINN-BTM in different time blocks. NN architecture and simulation parameters follow those of Fig. 16.

where v is defined by equation (77a). In light of (55)– (56), we set the loss function for HLConcPINN-ExBTM as follows,

$$Loss_i^I = \frac{W_1}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - v_{\theta_i}(x_{int}^n, t_{int}^n)\right]^2 + \frac{W_2}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial v_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) - \frac{\partial^2 u_{\theta_i}}{\partial x^2}(x_{int}^n, t_{int}^n) + u_{\theta_i}(x_{int}^n, t_{int}^n) + \sin(u_{\theta_i}(x_{int}^n, t_{int}^n)) - f(x_{int}^n, t_{int}^n)\right]^2$$

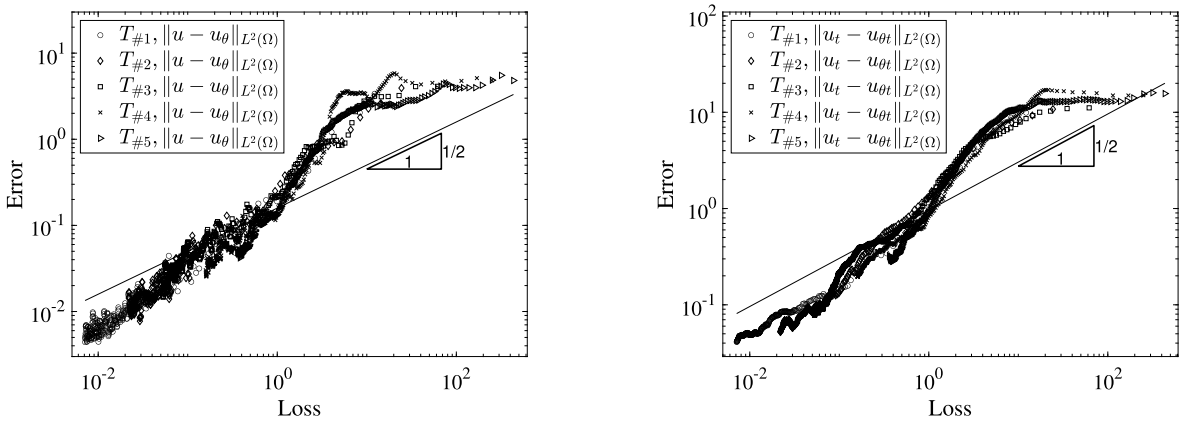


Fig. 18. Nonlinear Klein-Gordon equation: The l^2 errors of (a) u and (b) v as a function of the training loss value for the HLConcPINN-ExBTM method. The NN architecture and simulation parameters follow those of Fig. 16.

$$
\begin{aligned}
&+\frac{W_3}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial^2 u_{\theta_i}}{\partial t\partial x}(x_{int}^n,t_{int}^n)-\frac{\partial v_{\theta_i}}{\partial x}(x_{int}^n,t_{int}^n)\right]^2+\frac{W_4}{N_c}\sum_{j=1}^{i}\sum_{n=1}^{N_c}\left[u_{\theta_i}(x_{tb}^n,t_{j-1})-u_{\theta_{j-1}}(x_{tb}^n,t_{j-1})\right]^2\\
&+\frac{W_5}{N_c}\sum_{j=1}^{i}\sum_{n=1}^{N_c}\left[v_{\theta_i}(x_{tb}^n,t_{j-1})-v_{\theta_{j-1}}(x_{tb}^n,t_{j-1})\right]^2+\frac{W_6}{N_c}\sum_{j=1}^{i}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial x}(x_{tb}^n,t_{j-1})-\frac{\partial u_{\theta_{j-1}}}{\partial x}(x_{tb}^n,t_{j-1})\right]^2\\
&+W_7\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}\left[(v_\theta(0,t_{sb}^n)-\frac{\partial\phi_1}{\partial t}(t_{sb}^n))^2+(v_\theta(1,t_{sb}^n)-\frac{\partial\phi_2}{\partial t}(t_{sb}^n))^2\right]\Big)^{1/2}+Loss_{i-1}^I,
\end{aligned}
\tag{78}
$$

where W_i $(i=1,\cdots,7)$ are the penalty coefficients for different loss terms. The loss function for HLConcPINN-BTM is set to,

$$
\begin{aligned}
Loss_i^{II} =&\frac{W_1}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial t}(x_{int}^n,t_{int}^n)-v_{\theta_i}(x_{int}^n,t_{int}^n)\right]^2\\
&+\frac{W_2}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial v_{\theta_i}}{\partial t}(x_{int}^n,t_{int}^n)-\frac{\partial^2 u_{\theta_i}}{\partial x^2}(x_{int}^n,t_{int}^n)+u_{\theta_i}(x_{int}^n,t_{int}^n)+\sin(u_{\theta_i}(x_{int}^n,t_{int}^n))-f(x_{int}^n,t_{int}^n)\right]^2\\
&+\frac{W_3}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial^2 u_{\theta_i}}{\partial t\partial x}(x_{int}^n,t_{int}^n)-\frac{\partial v_{\theta_i}}{\partial x}(x_{int}^n,t_{int}^n)\right]^2+\frac{W_4}{N_c}\sum_{n=1}^{N_c}\left[u_{\theta_i}(x_{tb}^n,t_{i-1})-u_{\theta_{i-1}}(x_{tb}^n,t_{i-1})\right]^2\\
&+\frac{W_5}{N_c}\sum_{n=1}^{N_c}\left[v_{\theta_i}(x_{tb}^n,t_{i-1})-v_{\theta_{i-1}}(x_{tb}^n,t_{i-1})\right]^2+\frac{W_6}{N_c}\sum_{n=1}^{N_c}\left[\frac{\partial u_{\theta_i}}{\partial x}(x_{tb}^n,t_{i-1})-\frac{\partial u_{\theta_{i-1}}}{\partial x}(x_{tb}^n,t_{i-1})\right]^2\\
&+W_7\Big(\frac{1}{N_c}\sum_{n=1}^{N_c}\left[(v_\theta(0,t_{sb}^n)-\frac{\partial\phi_1}{\partial t}(t_{sb}^n))^2+(v_\theta(1,t_{sb}^n)-\frac{\partial\phi_2}{\partial t}(t_{sb}^n))^2\right]\Big)^{1/2}.
\end{aligned}
\tag{79}
$$

We employ the following values for the penalty coefficients, $(W_1,\cdots,W_7)=(0.4,0.4,0.4,0.6,0.6,0.6,0.6)$, for this problem. Five uniform time blocks are used in block time marching.

Figs. 16 and 17 provide an overview of the simulation results obtained by HLConcPINN-ExBTM and HLConcPINN-BTM for the nonlinear Klein-Gordon equation. Here the distributions of the HLConcPINN-ExBTM and HLConcPINN-BTM solutions for u and $v=\frac{\partial u}{\partial t}$, their point-wise absolute errors, as well as the exact solution field, have been shown. The loss histories for different time blocks obtained using these methods are shown in Fig. 17. The network architecture (consisting of three hidden layers), the activation functions, and the training collocation points are given in the caption of Fig. 16. The simulation results obtained with HLConcPINN-ExBTM are markedly more accurate than those of HLConcPINN-BTM for this problem, especially at later time (the last time block). It is also noted that the solution accuracy for $\frac{\partial u}{\partial t}$ is notably lower than that of u.

Table 10 summarizes a study of the training collocation points on the PINN solutions. We list the l^2 and l^∞ errors of both HLConcPINN-ExBTM and HLConcPINN-BTM in different time blocks obtained with a range of training collocation points between $N_c=1500$ and $N_c=3000$. The neural network architecture and activation functions follow those of Fig. 16. The results are in general not sensitive to the number of collocation points, similar to what has been obtained with other test problems in previous subsections.

Table 11 compares the simulation results of HLConcPINN-ExBTM and HLConcPINN-BTM obtained with different activation functions (tanh, Gaussian, swish, softplus) for the last hidden layer. Three hidden layers are employed in the neural network, with tanh

Table 10
Nonlinear Klein-Gordon equation: l^2 and l^∞ errors of u for HLConcPINN-ExBTM and HLConcPINN-BTM obtained with different training collocation points N_c. The NN architecture and activation functions follow those of Fig. 16.

Error	Time block	$N_c = 1500$		$N_c = 2000$		$N_c = 2500$		$N_c = 3000$	
		ExBTM	BTM	ExBTM	BTM	ExBTM	BTM	ExBTM	BTM
l^2	$T_{\#1}$	1.14e-03	1.52e-03	1.37e-03	1.64e-03	1.21e-03	1.52e-03	1.88e-03	1.56e-03
	$T_{\#2}$	3.08e-03	3.99e-03	3.28e-03	2.97e-03	3.94e-03	3.53e-03	6.14e-03	2.55e-03
	$T_{\#3}$	5.44e-03	9.56e-03	7.79e-03	6.89e-03	4.89e-03	6.96e-03	6.61e-03	7.75e-03
	$T_{\#4}$	9.69e-03	1.67e-02	1.90e-02	7.19e-03	8.31e-03	1.44e-02	1.01e-02	1.36e-02
	$T_{\#5}$	7.03e-02	9.11e-02	3.23e-02	1.95e-02	1.33e-02	6.49e-01	2.93e-02	7.43e-02
l^∞	$T_{\#1}$	3.33e-03	3.24e-03	5.01e-03	4.75e-03	3.62e-03	4.21e-03	4.40e-03	4.54e-03
	$T_{\#2}$	8.52e-03	8.47e-03	9.59e-03	1.07e-02	1.06e-02	9.18e-03	1.27e-02	7.50e-03
	$T_{\#3}$	1.93e-02	2.74e-02	1.77e-02	1.78e-02	1.44e-02	2.25e-02	1.89e-02	2.10e-02
	$T_{\#4}$	2.34e-02	4.90e-02	5.88e-02	1.57e-02	2.38e-02	4.62e-02	2.76e-02	3.37e-02
	$T_{\#5}$	1.89e-01	2.40e-01	9.73e-02	4.58e-02	4.25e-02	1.41e+00	8.66e-02	2.03e-01

Table 11
Nonlinear Klein-Gordon equation: l^2 and l^∞ errors of u for HLConcPINN-ExBTM and HLConcPINN-BTM obtained with different activation functions for the last hidden layer. NN: [2,90,90,10,2], with tanh activation function for the first two hidden layers and the activation function in the last hidden layer varied; $N_c = 2500$ for the training collocation points.

Error	Time block	tanh		Gaussian		Swish		Softplus	
		ExBTM	BTM	ExBTM	BTM	ExBTM	BTM	ExBTM	BTM
l^2	$T_{\#1}$	2.28e-03	2.86e-03	2.52e-03	3.41e-03	9.61e-04	1.98e-03	1.81e-03	1.40e-03
	$T_{\#2}$	5.24e-03	5.83e-03	1.82e-03	6.29e-03	4.54e-03	3.62e-03	6.21e-03	7.67e-03
	$T_{\#3}$	8.22e-03	1.68e-02	1.11e-02	1.97e-02	7.75e-03	5.16e-03	8.18e-03	1.01e-02
	$T_{\#4}$	1.73e-02	2.52e-02	3.25e-01	2.86e-01	1.13e-02	1.17e-02	1.72e-02	8.91e-03
	$T_{\#5}$	1.33e-01	8.68e-02	5.56e-01	7.53e-01	1.11e-01	1.82e-02	2.98e-02	1.43e-02
l^∞	$T_{\#1}$	5.73e-03	8.85e-03	5.89e-03	8.22e-03	3.30e-03	5.00e-03	5.54e-03	4.63e-03
	$T_{\#2}$	1.64e-02	1.74e-02	5.34e-03	1.94e-02	1.15e-02	1.61e-02	1.99e-02	2.67e-02
	$T_{\#3}$	2.24e-02	4.67e-02	2.87e-02	4.05e-02	1.49e-02	1.60e-02	2.20e-02	2.91e-02
	$T_{\#4}$	4.27e-02	7.18e-02	1.05e+00	7.84e-01	2.53e-02	3.32e-02	4.77e-02	2.33e-02
	$T_{\#5}$	3.65e-01	2.47e-01	1.43e+00	1.80e+00	3.42e-01	6.28e-02	8.76e-02	4.46e-02

activation for the first two hidden layers and the activation function of the last hidden layer varied. The network architecture and other simulation parameters are specified in the table caption. These results can be compared with that of Table 10 corresponding to $N_c = 2500$, where the sine activation function has been used for the last hidden layer. Among the activation functions tested, the sine function appears to yield the best simulation results.

Finally Fig. 18 illustrates the relation between the errors for u and v and the training loss for the HLConcPINN-ExBTM method from our simulations. The simulation data signify a scaling with a power of approximately $1/2$, which is roughly consistent with the conclusion of Theorem 6.4.

8. Concluding remarks

We have presented a hidden-layer concatenated physics informed neural network (HLConcPINN) method for approximating PDEs, by combining hidden-layer concatenated feed-forward neural networks (HLConcFNN), an extended block time marching strategy, and the physics informed approach. We analyze the convergence properties and the errors of this method for parabolic and hyperbolic type PDEs. Our analyses show that with this method the approximation error of the solution field can be effectively controlled by the training loss for dynamic simulations with long time horizons. HLConcPINN allows network architectures with an arbitrary number of hidden layers of two or larger, and any of the commonly-used smooth activation functions for all hidden layers beyond the first two. Our method generalizes several existing PINN techniques, which have theoretical guarantees but are confined to network architectures with two hidden layers and the tanh activation function. We implement the HLConcPINN algorithm, and have presented a number of computational examples based on this method. The numerical results demonstrate the effectiveness of our method and corroborate the relationship between the approximation error and the training loss function from theoretical analyses.

For long-term dynamic simulations, the current work adopts the block time marching (BTM) strategy. The original block time marching scheme, however, is not amenable to theoretical analysis, due to the difficulty caused by the regularity of the initial conditions on time blocks beyond the first one. The extended block time marching (ExBTM) strategy presented in this work circumvents this issue, by using the true initial data of the problem as the initial value for all time blocks and enforcing the residuals at a set of discrete temporal points from the preceding time blocks. This modified BTM strategy is crucial to the theoretical analysis of HLConcPINN for long-time dynamic simulations. A practical question when using BTM is how to choose the number of time blocks for a given problem and a given temporal dimension one would like to cover in the simulation. Our past experiences with the original BTM and the current modified BTM methods suggest that with smaller time blocks the network training tends to be easier and it

would generally produce more accurate results. On the other hand, with a fixed overall temporal dimension of the problem domain, increasing the number of time blocks (i.e. reducing the time block size) would increase the overall computation burden. We find that a moderate time block size is usually preferable and would typically suffice for many problems.

Finally we would like to comment that the analysis of the HLConcPINN technique, excluding the block time marching component, can be extended to elliptic type equations. An analysis of the nonlinear Helmholtz equation as an example of this type of equation is provided in the Appendix (Section 9.3).

Acknowledgments

The work was partially supported by the NSF of China (No. 12401540 and No. 12101495), China Postdoctoral Science Foundation (No. 2024M762629), Shanxi Postdoctoral Science Foundation (No. 2024BSHSDZZ166), Natural Science Foundation of Hunan Province (No. 2022JJ40422), and the US National Science Foundation (DMS-2012415).

9. Appendix: auxiliary results and proofs of main theorems

9.1. Some auxiliary results

Let a d-tuple of non-negative integers $\alpha \in \mathbb{N}_0^d$ be multi-index with $d \in \mathbb{N}$. For given two multi-indices $\alpha, \beta \in \mathbb{N}_0^d$, we say that $\alpha \le \beta$, if and only if, $\alpha_i \le \beta_i$ for all $i = 1, \cdots, d$. Let $\prod$ denote the product operator, representing the multiplication of a sequence of terms. Denote $|\alpha| = \sum_{i=1}^d \alpha_i$, $\alpha! = \prod_{i=1}^d \alpha_i!$, $\binom{\alpha}{\beta} = \frac{\alpha!}{\beta!(\alpha-\beta)!}$ with $\beta \le \alpha$. Let $P_{m,n} = \{\alpha \in \mathbb{N}_0^n, |\alpha| = m\}$, for which it holds $|P_{m,n}| = \binom{m+n-1}{m}$.

Lemma 9.1. *Let $d \in \mathbb{N}, k \in \mathbb{N}_0$, $f \in H^k(\Omega)$ and $g \in W^{k,\infty}(\Omega)$ with $\Omega \subset \mathbb{R}^d$, then $\|fg\|_{H^k(\Omega)} \le 2^k \|f\|_{H^k(\Omega)} \|g\|_{W^{k,\infty}(\Omega)}$.*

Lemma 9.2 *(Multiplicative trace inequality, e.g. [12]). Let $d \ge 2$, $\Omega \subset \mathbb{R}^d$ be a Lipschitz domain and let $\gamma_0 : H^1(\Omega) \to L^2(\partial\Omega) : u \mapsto u|_{\partial\Omega}$ be the trace operator. Denote by h_Ω the diameter of Ω and by ρ_Ω the radius of the largest d-dimensional ball that can be inscribed into Ω. Then it holds that*

$$\|\gamma_0 u\|_{L^2(\partial\Omega)} \le C_{h_\Omega,d,\rho_\Omega} \|u\|_{H^1(\Omega)}, \quad \text{where } C_{h_\Omega,d,\rho_\Omega} = \sqrt{2\max\{2h_\Omega, d\}/\rho_\Omega}. \tag{80}$$

Lemma 9.3 *([12]). Let $d, n, L, W \in \mathbb{N}$ and let $u_\vartheta : \mathbb{R}^d \to \mathbb{R}^{l_L}$ be a neural network with $\vartheta \in \Theta$ for $d, L \ge 2, R, W \ge 1$, cf. Definition 2.1. Assume that $\|\sigma\|_{C^n} \ge 1$. Then it holds for $1 \le j \le l_L$ that*

$$\|(u_\vartheta)_j\|_{C^n(\Omega)} \le 16^L d^{2n} (e^2 n^4 W^3 R^n \|\sigma\|_{C^n(\Omega)})^{nL}. \tag{81}$$

Remark 9.4. Let $u_\theta : \mathbb{R}^d \to \mathbb{R}^{l_L}$ denote a neural network with smooth activation functions, in accordance with Definition 2.1. Suppose the first two hidden layers of the network are endowed with the tanh activation function, whereas (if $L > 3$) the subsequent hidden layers utilize a variety of smooth activation functions, including (but not restricted to) e.g. the tanh, sine, sigmoid, Gaussian, and softplus functions. Let $\hat{\sigma}$ denote a collection of these smooth activation functions. Under the conditions specified in Lemma 9.3, by defining $\|\sigma\|_{C^k} = \max_{\tilde{\sigma}\in\hat{\sigma}}\{\|\tilde{\sigma}\|_{C^k}\}$, it can be shown that Lemma 9.3 remains valid. Furthermore, thanks to the inherent properties of hidden-layer concatenated feedforward neural networks, the output fields of the i-th ($i = 1, \cdots, L-1$) hidden layer and the output layer exhibit analogous behavior based on Lemma 9.3. For the sake of conciseness, we omit the proof here and refer to the results presented in Lemma 9.3.

Lemma 9.5 *([12]). Let $d \ge 2, n \ge 2, m \ge 3$, $\delta > 0$, $a_i, b_i \in \mathbb{Z}$ with $a_i < b_i$ for $1 \le i \le d$, $\Omega = \prod_{i=1}^d [a_i, b_i]$ and $f \in H^m(\Omega)$. Then for every $N \in \mathbb{N}$ with $N > 5$, there exists a tanh neural network $\widetilde{f}^N$ with two hidden layers, one of widths at most $3\lceil \frac{m+n-2}{2} \rceil |P_{m-1,d+1}| + \sum_{i=1}^d (b_i - a_i)(N-1)$ and another of width at most $3\lceil \frac{d+n}{2} \rceil |P_{d+1,d+1}| N^d \prod_{i=1}^d (b_i - a_i)$, such that for $k = 0, 1, 2$ it holds that*

$$\|f - \widetilde{f}^N\|_{H^k(\Omega)} \le 2^k 3^d C_{k,m,d,f} (1+\delta) \ln^k \left(\beta_{k,\delta,d,f} N^{d+m+2}\right) N^{-m+k}, \tag{82}$$

where

$$C_{k,m,d,f} = \max_{0\le l\le k} \binom{d+l-1}{l}^{1/2} \frac{((m-l)!)^{1/2}}{(\lceil \frac{m-l}{d} \rceil !)^{d/2}} \left(\frac{3\sqrt{d}}{\pi}\right)^{m-l} |f|_{H^m(\Omega)}, \tag{83}$$

$$\beta_{k,\delta,d,f} = \frac{5 \cdot 2^{kd} \max\{\prod_{i=1}^d (b_i - a_i), d\} \max\{\|f\|_{W^{k,\infty}(\Omega)}, 1\}}{3^d \delta \min\{1, C_{k,m,d,f}\}}, \tag{84}$$

and $\lceil \cdot \rceil$ denotes the ceiling function, which maps a real number to the smallest integer greater than or equal to it. Moreover, the weights of $\widetilde{f}^N$ scale as $\mathcal{O}(N^\gamma + N \ln N)$ with $\gamma = \max\{m^2/n, d(2+m+d)/n\}$.

Lemma 9.6 *([38]). Given an architectural vector $\boldsymbol{l}_1 = (l_0, l_1, \cdots, l_{L-1}, l_L)$ with $l_L = 1$, define a new vector $\boldsymbol{l}_2 = (l_0, l_1, \cdots, l_{L-1}, n, l_L)$, where $n \geq 1$ is an integer. For a given domain $D \subset \mathbb{R}^{l_0}$ and an activation function σ, the following relation holds*

$$U(D, \boldsymbol{l}_1, \sigma) \subseteq U(D, \boldsymbol{l}_2, \sigma), \tag{85}$$

where U is defined by (5).

Lemma 9.7 *([38]). Given an architectural vector $\boldsymbol{l}_1 = (l_0, l_1, \cdots, l_{L-1}, l_L)$ with $l_L = 1$, define a new vector $\boldsymbol{l}_2 = (l_0, l_1, \cdots, l_{s-1}, l_s + 1, l_{s+1}, \cdots, l_L)$ for some s $(1 \leq s \leq L-1)$. For a given domain $D \subset \mathbb{R}^{l_0}$ and an activation function σ, the following relation holds*

$$U(D, \boldsymbol{l}_1, \sigma) \subseteq U(D, \boldsymbol{l}_2, \sigma), \tag{86}$$

where U is defined by (5).

Lemma 9.8. *Under the conditions specified in Lemma 9.5, for every $N \in \mathbb{N}$ where $N > 5$, there exists a hidden-layer concatenated feedforward neural network denoted as $\hat{f}^N$, defined by*

$$\hat{f}^N = \sum_{i=1}^{L-1} M_i \hat{f}_i^N + b_L \qquad L \geq 3, \tag{87}$$

where $\hat{f}_i^N$, $M_i \in \mathbb{R}^{1 \times l_i}$ $(1 \leq i \leq L-1)$ and $b_L \in \mathbb{R}^1$ represent the output of the i-th hidden layer, the connection coefficients between the output layer and the i-th hidden layer, and the bias of the output layer, respectively. Note that the first two hidden layers of the network employ the tanh activation function, while the other hidden layers can use any other smooth activation function. For $k = 0, 1, 2$, this neural network satisfies

$$\|f - \hat{f}^N\|_{H^k(\Omega)} \lesssim \ln^k\left(N^{d+m+2}\right) N^{-m+k}. \tag{88}$$

Proof. Lemmas 9.6 and 9.7 imply that $\hat{f}^N$ possesses a greater representational capacity compared to $\widetilde{f}^N$.

It should be noted that smooth functions are both continuous and bounded on a closed interval. For neural networks, activation function σ such as the sigmoid and hyperbolic tangent (tanh) are examples of smooth functions. These functions can be bounded by a constant C in the $H^k(\Omega)$ norm, i.e., $\|\sigma\|_{H^k(\Omega)} \leq C$.

We set $M_1 = 0^{1 \times l_1} \in \mathbb{R}^{1 \times l_1}$, $M_2 = W_3$, $b_L = b_3$, and $M_i = \frac{\epsilon}{C l_i (L-3)} 1^{1 \times l_i} \in \mathbb{R}^{1 \times l_i}$ $(i = 3, \cdots, L-1)$, while assigning $0^{l_i \times l_{i-1}} \in \mathbb{R}^{l_i \times l_{i-1}}$ to the weight coefficients W_i of the i-th hidden layer for all $i = 3, \cdots, L-1$. By retaining the initial two hidden layers in Lemma 9.5 and setting $\epsilon = \ln^k\left(N^{d+m+2}\right) N^{-m+k}$, we obtain $W_3 \hat{f}_2^N + b_3 = \widetilde{f}^N$ (defined in Lemma 9.5) and $\hat{f}^N = \widetilde{f}^N + \sum_{i=3}^{L-1} M_i \sigma(b_i)$. Consequently, the approximation can be bounded as follows

$$\|f - \hat{f}^N\|_{H^k(\Omega)} \leq \|f - \widetilde{f}^N\|_{H^k(\Omega)} + \sum_{i=3}^{L-1} \|M_i \sigma(b_i)\|_{H^k(\Omega)} \lesssim \ln^k\left(N^{d+m+2}\right) N^{-m+k},$$

where l_i denotes the number of nodes in the i-th hidden layer. □

9.2. Proof of main theorems from Section 3: heat equation

Proof of Theorem 3.3 :

Proof. Based on Lemma 9.8, there exists a HLConcPINN u_{θ_i} such that for every $0 \leq m \leq 2$,

$$\|u_{\theta_i} - u\|_{H^m(\widetilde{\Omega}_i)} \lesssim N^{-k+m} \ln^m(N). \tag{89}$$

According to Lemma 9.2, we can bound the HLConcPINN residual terms,

$$\|\frac{\partial \hat{u}_i}{\partial t}\|_{L^2(\widetilde{\Omega}_i)} \leq \|\hat{u}_i\|_{H^1(\widetilde{\Omega}_i)}, \qquad \|\Delta \hat{u}_i\|_{L^2(\widetilde{\Omega}_i)} \leq \|\hat{u}_i\|_{H^2(\widetilde{\Omega}_i)},$$
$$\|\hat{u}_i\|_{L^2(\widetilde{\Omega}_{si})} \leq \|\hat{u}_i\|_{L^2(\partial\widetilde{\Omega}_i)} \leq C_{h_{\widetilde{\Omega}_i}, d+1, \rho_{\widetilde{\Omega}_i}} \|\hat{u}_i\|_{H^1(\widetilde{\Omega}_i)}.$$

For $j = 1$, $R_{tb_i}|_{t=t_0} = \hat{u}_i|_{t=t_0}$, we obtain

$$\|R_{tb_i}(\boldsymbol{x}, 0)\|_{L^2(D)} \leq \|\hat{u}_i\|_{L^2(\partial\widetilde{\Omega}_i)} \leq C_{h_{\widetilde{\Omega}_i}, d+1, \rho_{\widetilde{\Omega}_i}} \|\hat{u}_i\|_{H^1(\widetilde{\Omega}_i)}.$$

For $j > 1$, it holds

$$\begin{aligned}\|R_{tb_i}(\boldsymbol{x}, t_{j-1})\|_{L^2(D)} &\leq \|\hat{u}_i|_{t=t_{j-1}}\|_{L^2(D)} + \|\hat{u}_{j-1}|_{t=t_{j-1}}\|_{L^2(D)} \leq \|\hat{u}_i\|_{L^2(\partial\widetilde{\Omega}_{j-1})} + \|\hat{u}_{j-1}\|_{L^2(\partial\widetilde{\Omega}_{j-1})} \\ &\leq C_{h_{\widetilde{\Omega}_{j-1}}, d+1, \rho_{\widetilde{\Omega}_{j-1}}} (\|\hat{u}_i\|_{H^1(\widetilde{\Omega}_{j-1})} + \|\hat{u}_{j-1}\|_{H^1(\widetilde{\Omega}_{j-1})}) \leq C_{h_{\widetilde{\Omega}_{j-1}}, d+1, \rho_{\widetilde{\Omega}_{j-1}}} (\|\hat{u}_i\|_{H^1(\widetilde{\Omega}_i)} + \|\hat{u}_{j-1}\|_{H^1(\widetilde{\Omega}_{j-1})}).\end{aligned}$$

By combining these relations with (89), we can obtain

$$\|R_{int_i}\|_{L^2(\widetilde{\Omega}_i)} = \|\frac{\partial \hat{u}_i}{\partial t} - \Delta \hat{u}_i\|_{L^2(\widetilde{\Omega}_i)} \leq \|\hat{u}_i\|_{H^1(\widetilde{\Omega}_i)} + \|\hat{u}_i\|_{H^2(\widetilde{\Omega}_i)} \lesssim N^{-k+2}\ln^2 N,$$
$$\|R_{tb_i}(\boldsymbol{x}, t_{j-1})\|_{L^2(D)}, \|R_{sb_i}\|_{L^2(\widetilde{\Omega}_{si})} \lesssim \|\hat{u}_i\|_{H^1(\widetilde{\Omega}_i)} + \|\hat{u}_{j-1}\|_{H^1(\widetilde{\Omega}_{j-1})} \lesssim N^{-k+1}\ln N \quad 1 \leq j \leq i.$$

Then, we finish our proof. □

Proof of Theorem 3.4 :

Proof. Take the inner product of (14a) and $\hat{u}_i$ over D to obtain,

$$\begin{aligned}\frac{d}{2dt}\int_D |\hat{u}_i|^2 \mathrm{d}\boldsymbol{x} &= -\int_D |\nabla \hat{u}_i|^2 \mathrm{d}\boldsymbol{x} + \int_{\partial D} R_{sb_i} \nabla \hat{u}_i \cdot \boldsymbol{n} \mathrm{d}s(\boldsymbol{x}) + \int_D R_{int_i}\hat{u}_i \mathrm{d}\boldsymbol{x} \\ &\leq -\int_D |\nabla \hat{u}_i|^2 \mathrm{d}\boldsymbol{x} + \frac{1}{2}\int_D |\hat{u}_i|^2 \mathrm{d}\boldsymbol{x} + \frac{1}{2}\int_D |R_{int_i}|^2 \mathrm{d}\boldsymbol{x} + C_{\partial D_i}\Big(\int_{\partial D} |R_{sb_i}|^2 \mathrm{d}s(\boldsymbol{x})\Big)^{\frac{1}{2}},\end{aligned} \tag{90}$$

where $C_{\partial D_i} = |\partial D|^{\frac{1}{2}}(\|u\|_{C^1(\partial D\times[t_{i-1},t_i])} + \|u_{\theta_i}\|_{C^1(\partial D\times[t_{i-1},t_i])})$.

Integrating (90) over $[t_{i-1}, \tau]$ for any $t_{i-1} < \tau \leq t_i$, using the initial condition (14b), and applying Cauchy–Schwarz inequality, we obtain

$$\begin{aligned}&\int_D |\hat{u}_i(\boldsymbol{x},\tau)|^2 \mathrm{d}\boldsymbol{x} + 2\int_{t_{i-1}}^{t_i}\int_D |\nabla \hat{u}_i|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\ &\leq \int_D |\hat{u}_i(\boldsymbol{x},t_{i-1})|^2 \mathrm{d}\boldsymbol{x} + \int_{t_{i-1}}^{t_i}\int_D |\hat{u}_i|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t + \int_{t_{i-1}}^{t_i}\int_D |R_{int_i}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t + 2C_{\partial D_i}|\Delta t|^{\frac{1}{2}}\Big(\int_{t_{i-1}}^{t_i}\int_{\partial D} |R_{sb_i}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t\Big)^{\frac{1}{2}} \\ &\leq \int_D |\hat{u}_i(\boldsymbol{x},t_{i-1})|^2 \mathrm{d}\boldsymbol{x} + \sum_{j=1}^{i-1}\int_D |R_{tb_i}(\boldsymbol{x},t_{j-1})|^2 \mathrm{d}\boldsymbol{x} + \int_{t_{i-1}}^{t_i}\int_D |\hat{u}_i|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t + \int_{t_{i-1}}^{t_i}\int_D |R_{int_i}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\ &\quad + 2C_{\partial D_i}|\Delta t|^{\frac{1}{2}}\Big(\int_{t_{i-1}}^{t_i}\int_{\partial D} |R_{sb_i}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t\Big)^{\frac{1}{2}} \\ &\leq 2\int_D |\hat{u}_{i-1}(\boldsymbol{x},t_{i-1})|^2 \mathrm{d}\boldsymbol{x} + 2\int_D |R_{tb_i}(\boldsymbol{x},t_{i-1})|^2 \mathrm{d}\boldsymbol{x} + \sum_{j=1}^{i-1}\int_D |R_{tb_i}(\boldsymbol{x},t_{j-1})|^2 \mathrm{d}\boldsymbol{x} + \int_{t_{i-1}}^{t_i}\int_D |\hat{u}_i|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\ &\quad + \int_{t_{i-1}}^{t_i}\int_D |R_{int_i}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t + 2C_{\partial D_i}|\Delta t|^{\frac{1}{2}}\Big(\int_{t_{i-1}}^{t_i}\int_{\partial D} |R_{sb_i}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t\Big)^{\frac{1}{2}}.\end{aligned}$$

Then, applying the integral form of the Grönwall inequality to the above inequality, it holds

$$\begin{aligned}&\int_D |\hat{u}_i(\boldsymbol{x},\tau)|^2 \mathrm{d}\boldsymbol{x} + 2\int_{t_{i-1}}^{t_i}\int_D |\nabla \hat{u}_i|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\ &\leq \Big(2\int_D |\hat{u}_{i-1}(\boldsymbol{x},t_{i-1})|^2 \mathrm{d}\boldsymbol{x} + 2\sum_{j=1}^{i}\int_D |R_{tb_i}(\boldsymbol{x},t_{j-1})|^2 \mathrm{d}\boldsymbol{x} + \int_{t_{i-1}}^{t_i}\int_D |R_{int_i}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t\Big)\exp(\Delta t) \\ &\quad + 2C_{\partial D_i}|\Delta t|^{\frac{1}{2}}\Big(\int_{t_{i-1}}^{t_i}\int_{\partial D} |R_{sb_i}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t\Big)^{\frac{1}{2}}\exp(\Delta t).\end{aligned} \tag{91}$$

Firstly, by (91) and integrating (91) over $[t_0, t_1]$, Theorem 3.4 holds for $i = 1$ according to the fact that $\mathcal{E}_{G_{i-1}}(\theta) = 0$ and $\hat{u}_{i-1}|_{t=t_{i-1}} = 0$ for $i = 1$.

Secondly, we assume that Theorem 3.4 holds for all $i \le l-1$, i.e.,

$$\int_D |\hat{u}_i(\boldsymbol{x},\tau)|^2 \mathrm{d}\boldsymbol{x} \le C_{G_i} \exp(\Delta t), \qquad \int_{t_{i-1}}^{t_i} \int_D |\hat{u}_i(\boldsymbol{x},\tau)|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \le C_{G_i} \Delta t \exp(\Delta t).$$

For $i = l-1$, as it should be,

$$\int_D |\hat{u}_{l-1}(\boldsymbol{x},\tau)|^2 \mathrm{d}\boldsymbol{x} \le C_{G_{l-1}} \exp(\Delta t) \qquad t_{l-2} \le \tau \le t_{l-1}. \tag{92}$$

Finally, we begin to verify that Theorem 3.4 is true at $i = l$.

Let $i = l$ in (91), under the established conditions (92) and the Grönwall inequality, we derive

$$\begin{aligned}
&\int_D |\hat{u}_l(\boldsymbol{x},\tau)|^2 \mathrm{d}\boldsymbol{x} + 2\int_{t_{l-1}}^{t_l} \int_D |\nabla \hat{u}_l|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\
&\le \Big(2\int_D |\hat{u}_{l-1}(\boldsymbol{x},t_{l-1})|^2 \mathrm{d}\boldsymbol{x} + 2\sum_{j=1}^{l} \int_D |R_{tb_l}(\boldsymbol{x},t_{j-1})|^2 \mathrm{d}\boldsymbol{x} + \int_{t_{l-1}}^{t_l} \int_D |R_{int_l}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t\Big) \exp(\Delta t) \\
&\quad + 2C_{\partial D_l} |\Delta t|^{\frac{1}{2}} \Big(\int_{t_{l-1}}^{t_l} \int_{\partial D} |R_{sb_l}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t\Big)^{\frac{1}{2}} \exp(\Delta t) \\
&\le \Big(2C_{G_{l-1}} \exp(\Delta t) + 2\sum_{j=1}^{l} \int_D |R_{tb_l}(\boldsymbol{x},t_{j-1})|^2 \mathrm{d}\boldsymbol{x} + \int_{t_{l-1}}^{t_l} \int_D |R_{int_l}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t\Big) \exp(\Delta t) \\
&\quad + 2C_{\partial D_l} |\Delta t|^{\frac{1}{2}} \Big(\int_{t_{l-1}}^{t_l} \int_{\partial D} |R_{sb_l}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t\Big)^{\frac{1}{2}} \exp(\Delta t) \\
&\le (\widetilde{C}_{G_l} + 2C_{G_{l-1}} \exp(\Delta t)) \exp(\Delta t),
\end{aligned}$$

where

$$\widetilde{C}_{G_l} = 2\sum_{j=1}^{l} \int_D |R_{tb_l}(\boldsymbol{x},t_{j-1})|^2 \mathrm{d}\boldsymbol{x} + \int_{t_{l-1}}^{t_l} \int_D |R_{int_l}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t + 2C_{\partial D_l} |\Delta t|^{\frac{1}{2}} \Big(\int_{t_{l-1}}^{t_l} \int_{\partial D} |R_{sb_l}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t\Big)^{\frac{1}{2}}.$$

By using the mathematical induction and deduction methods, we finish the proof. □

Proof of Theorem 3.5 :

Proof. By combining Theorem 3.4 with the quadrature error formula (3), we have

$$\begin{aligned}
&\int_D |R_{tb_i}|^2 \mathrm{d}\boldsymbol{x} = \int_D |R_{tb_i}|^2 \mathrm{d}\boldsymbol{x} - Q^D_{M_{tb_i}}[R^2_{tb_i}] + Q^D_{M_{tb_i}}[R^2_{tb_i}] \le C_{(R^2_{tb_i})} M_{tb_i}^{-\frac{2}{d}} + Q^D_{M_{tb_i}}[R^2_{tb_i}], \\
&\int_{\Omega_i} |R_{int_i}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t = \int_{\Omega_i} |R_{int_i}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t - Q^{\Omega_i}_{M_{int_i}}[R^2_{int_i}] + Q^{\Omega_i}_{M_{int_i}}[R^2_{int_i}] \le C_{(R^2_{int_i})} M_{int_i}^{-\frac{2}{d+1}} + Q^{\Omega_i}_{M_{int_i}}[R^2_{int_i}], \\
&\int_{\Omega_{*i}} |R_{sb_i}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t = \int_{\Omega_{*i}} |R_{sb_i}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t - Q^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb_i}] + Q^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb_i}] \le C_{(R^2_{sb_i})} M_{sb_i}^{-\frac{2}{d}} + Q^{\Omega_{*i}}_{M_{sb_i}}[R^2_{sb_i}].
\end{aligned}$$

Combining the fact that $C_{(R^2_{tb_i})} \lesssim \|R^2_{tb_i}\|_{C^2}$ and $\|R^2_{tb_i}\|_{C^n} \le 2^n \|R_{tb_i}\|^2_{C^n}$ with Lemma 9.3, it holds

$$\begin{aligned}
C_{(R^2_{tb_i}(\boldsymbol{x},t_{j-1}))} &\lesssim \|R_{tb_i}(\boldsymbol{x},t_{j-1})\|^2_{C^2} \le 2(\|\hat{u}_i|_{t=t_{j-1}}\|^2_{C^2} + \|\hat{u}_{j-1}|_{t=t_{j-1}}\|^2_{C^2}) \\
&\lesssim \|u\|^2_{C^2} + (e^2 2^4 W^3 R^2 \|\sigma\|_{C^2})^{4L}.
\end{aligned} \tag{93}$$

In a similar way, we can estimate the terms $\int_{\Omega_i} |R_{int_i}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t$ and $\int_{\Omega_{*i}} |R_{sb_i}|^2 \mathrm{d}s(\boldsymbol{x})\mathrm{d}t$.

Then, combining the above inequalities with (17), it holds that

$$\int_{t_{i-1}}^{t_i}\int_D |\hat{u}_i(\boldsymbol{x},t)|^2 \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \le C_{T_i}\Delta t \exp(\Delta t),$$

where the constant C_{T_i} is defined in (20). □

9.3. HLConcPINN for approximating the nonlinear Helmholtz equation

9.3.1. Nonlinear Helmholtz equation

Let $D \subset \mathbb{R}^d$ $(d \ge 2)$ be an open connected bounded domain with a C^k boundary ∂D. We consider the following nonlinear Helmholtz equation:

$$-\Delta u(\boldsymbol{x}) + \lambda u(\boldsymbol{x}) + \alpha g(u(\boldsymbol{x})) = f(\boldsymbol{x}) \qquad \boldsymbol{x} \in D, \tag{94a}$$

$$u(\boldsymbol{x}) = u_d(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial D, \tag{94b}$$

where f is a source term, u_d denotes the boundary data, and the nonlinear term $g(u)$ is globally Lipschitz, i.e. there exists a constant L (independent of v and w) such that

$$|g(v) - g(w)| \le L|v - w| \quad \forall v, w \in \mathbb{R}. \tag{95}$$

Here, $\alpha \ge 0$ and $\lambda > 0$ are constants that satisfy $\lambda > 2L\alpha$.

9.3.2. Hidden-layer concatenated physics informed neural networks

We seek a HLConcPINN u_θ to approximate the solution u of (94) and define the following residuals:

$$R_{int}[u_\theta](\boldsymbol{x}) = -\Delta u_\theta + \lambda u_\theta + \alpha g(u_\theta) - f \qquad \boldsymbol{x} \in D, \tag{96a}$$

$$R_{sb}[u_\theta](\boldsymbol{x}) = u_\theta - u_d \qquad \boldsymbol{x} \in \partial D. \tag{96b}$$

Note that $R_{int}[u] = R_{sb}[u] = 0$ for the exact solution u.

With HLConcPINN we minimize the quantity,

$$\mathcal{E}_G(\theta)^2 = \int_D |R_{int}[u_\theta](\boldsymbol{x})|^2 \,\mathrm{d}\boldsymbol{x} + \left(\int_{\partial D} |R_{sb}[u_\theta](\boldsymbol{x})|^2 \,\mathrm{d}s(\boldsymbol{x})\right)^{\frac{1}{2}}. \tag{97}$$

For the nonlinear Helmholtz equation (94), we choose the training set $S \subset \overline{D}$ with $S = S_{int} \cup S_{sb}$, based on suitable quadrature points:

- Interior training points $S_{int} = \{\boldsymbol{x}_{int}^n \in D, 1 \le n \le N_{int}\}$.
- Spatial boundary training points $S_{sb} = \{\boldsymbol{x}_{sb}^n \in \partial D, 1 \le n \le N_{sb}\}$.

With the training set S, the integrals in (97) are approximated by a numerical quadrature, resulting in the training loss function,

$$\mathcal{E}_T(\theta, S)^2 = \mathcal{E}_T^{int}(\theta, S_{int})^2 + \mathcal{E}_T^{sb}(\theta, S_{sb}), \tag{98}$$

where

$$\mathcal{E}_T^{int}(\theta, S_{int})^2 = \sum_{n=1}^{N_{int}} \omega_{int}^n |R_{int}[u_\theta](\boldsymbol{x}_{int}^n)|^2, \qquad \mathcal{E}_T^{sb}(\theta, S_{sb})^2 = \sum_{n=1}^{N_{sb}} \omega_{sb}^n |R_{sb}[u_\theta](\boldsymbol{x}_{sb}^n)|^2, \tag{99}$$

with the data sets $S_{int} = \{\boldsymbol{x}_{int}^n\}_{n=1}^{N_{int}}$ and $S_{sb} = \{\boldsymbol{x}_{sb}^n\}_{n=1}^{N_{sb}}$ and the corresponding quadrature weights ω_{int}^n and ω_{sb}^n.

9.3.3. Error analysis

Let $\hat{u} = u_\theta - u$ denote the error of the HLConcPINN solution u_θ against the true solution u. Combining equation (94) and the definitions of different residuals (96), we have

$$R_{int} = -\Delta\hat{u} + \lambda\hat{u} + \alpha(g(u_\theta) - g(u)), \tag{100a}$$

$$R_{sb} = \hat{u}|_{\partial D}. \tag{100b}$$

Theorem 9.9. *Suppose n, d, $k \in N$ with $n, d \ge 2$ and $k \ge 3$, and $u \in H^k(D)$. For every integer $N > 5$, there exists a HLConcPINN u_θ, such that*

$$\|R_{int}\|_{L^2(D)} \lesssim N^{-k+2}\ln^2 N, \qquad \|R_{sb}\|_{L^2(\partial D)} \lesssim N^{-k+1}\ln N. \tag{101}$$

Proof. The conclusion follows from $u \in H^k(D)$, and the Lemmas 9.2 and 9.8. □

Theorem 9.10. *Let $d \in \mathbb{N}$ with $d \geq 2$, and $u \in C^1(D)$ be the classical solution to the nonlinear Helmholtz equation (94). Let u_θ denote a HLConcPINN with parameter θ. Then the following relation holds*

$$\int_D |\nabla \hat{u}|^2 \mathrm{d}\boldsymbol{x} + \left(\frac{\lambda}{2} - L\alpha\right) \int_D |\hat{u}|^2 \mathrm{d}\boldsymbol{x} \leq \frac{1}{2\lambda} \int_D |R_{int}|^2 \mathrm{d}\boldsymbol{x} + C_{\partial D} \left(\int_{\partial D} |R_{sb}|^2 \mathrm{d}s(\boldsymbol{x}) \right)^{\frac{1}{2}}, \tag{102}$$

where $C_{\partial D} = |\partial D|^{\frac{1}{2}} (\|u\|_{C^1(\partial D)} + ||u_\theta||_{C^1(\partial D)})$.

Proof. By following a similar approach to the proof of Theorem 3.4 and using the conditions (95) and $\lambda > 2L\alpha$, one can arrive at the relation (102). □

Theorem 9.11. *Let $d \in \mathbb{N}$ with $d \geq 2$, and $u \in C^4(D)$ be the classical solution of the nonlinear Helmholtz equation (94). Let u_θ be a HLConcPINN with parameter θ. Then the following relation holds,*

$$\begin{aligned} &\int_D |\nabla \hat{u}|^2 \mathrm{d}\boldsymbol{x} + \left(\frac{\lambda}{2} - L\alpha\right) \int_D |\hat{u}|^2 \mathrm{d}\boldsymbol{x} \leq \frac{1}{2\lambda} \left(C_{(R^2_{int})} M_{int}^{-\frac{2}{d}} + Q^D_{M_{int}}[R^2_{int}] \right) + C_{\partial D} \left(C_{(R^2_{sb})} M_{sb}^{-\frac{2}{d-1}} + Q^{\partial D}_{M_{sb}}[R^2_{sb}] \right)^{\frac{1}{2}} \\ &= \mathcal{O}\left(\mathcal{E}_T(\theta, S)^2 + M_{int}^{-\frac{2}{d}} + M_{sb}^{-\frac{1}{d-1}} \right), \end{aligned}$$

where $Q^D_{M_{int}}[R^2_{int}]$ and $Q^{\partial D}_{M_{sb}}[R^2_{sb}]$ represent the midpoint rule applied to the respective functions R^2_{int} and R^2_{sb}.

Proof. The result follows directly from Theorem 9.10, Lemma 9.3 and the quadrature error formula (3). □

Remark 9.12. For $\alpha = 0$, the nonlinear Helmholtz equation (94) reduces to a linear equation. The above analyses, including Theorems 9.9 to 9.11, all carry over to the linear case.

9.4. HLConcPINN for approximating the convection equation

9.4.1. Convection equation

Consider the following convection equation on the domain $D = [a, b] \subset \mathbb{R}^1$:

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial u(x,t)}{\partial x} = 0 \qquad (x,t) \in D \times [0,T], \tag{103a}$$

$$u(x,0) = u_{in}(x) \qquad x \in D, \tag{103b}$$

$$u(a,t) = g(t) \qquad t \in [0,T], \tag{103c}$$

where $g(t)$ denotes the boundary data and $u_{in}(x)$ is the initial distribution.

9.4.2. Hidden-layer concatenated physics informed neural networks

Based on the settings from Section 2.5, we seek HLConcPINN $u_{\theta_i} : D \times [0, t_i] \to \mathbb{R}$ for $1 \leq i \leq l$ (l denoting the number of time blocks) to approximate the solution u of (103). Define the following residual functions, for $1 \leq i \leq l$,

$$R_{int_i}[u_{\theta_i}](x,t) = \frac{\partial u_{\theta_i}}{\partial t} + \frac{\partial u_{\theta_i}}{\partial x}, \tag{104a}$$

$$R_{tb_i}[u_{\theta_i}](x) = u_{\theta_i}|_{t=t_{j-1}} - u_{\theta_{j-1}}|_{t=t_{j-1}} \qquad 1 \leq j \leq i, \tag{104b}$$

$$R_{sb_i}[u_{\theta_i}](t) = u_{\theta_i}(a,t) - g(t). \tag{104c}$$

In these equations $u_{\theta_0}\big|_{t=t_0} = u_{in}(x)$. Note that $R_{int_i}[u] = R_{tb_i}[u] = R_{sb_i}[u] = 0$ for the exact solution u. With HLConcPINN we determine θ_i ($1 \leq i \leq l$) by minimizing the following quantities,

$$\mathcal{E}_{G_i}(\theta_i)^2 = \widetilde{\mathcal{E}}_{G_i}(\theta_i)^2 + \mathcal{E}_{G_{i-1}}(\theta_{i-1})^2 \quad 1 \leq i \leq l, \tag{105}$$

$$\widetilde{\mathcal{E}}_{G_i}(\theta_i)^2 = \int_{t_{i-1}}^{t_i} \int_D |R_{int_i}[u_{\theta_i}](x,t)|^2 \mathrm{d}x \mathrm{d}t + \int_{t_{i-1}}^{t_i} |R_{sb_i}[u_{\theta_i}](a,t)|^2 \mathrm{d}t + \sum_{j=1}^{i} \int_D |R_{tb_i}[u_{\theta_i}](x,t_{j-1})|^2 \mathrm{d}x, \tag{106}$$

where $\mathcal{E}_{G_{i-1}}(\theta_{i-1}) = 0$ for $i = 1$.

The training set consists of $\mathcal{S} = \bigcup_{i=1}^{l} \mathcal{S}_i$ with $\mathcal{S}_i = \mathcal{S}_{int_i} \cup \mathcal{S}_{sb_i} \cup \mathcal{S}_{tb_i}$. The spatial boundary training points are $\mathcal{S}_{sb_i} = \{y_n\}$ for $1 \le n \le N_{sb_i}$, with $y_n = (x,t)_n \in \{a\} \times (t_{i-1}, t_i)$. We approximate the integrals in (105) by the mid-point rule, leading to the training loss functions,

$$\mathcal{E}_{T_i}(\theta_i, \mathcal{S}_i)^2 = \widetilde{\mathcal{E}}_{T_i}(\theta_i, \mathcal{S}_i)^2 + \mathcal{E}_{T_{i-1}}(\theta_{i-1}, \mathcal{S}_{i-1})^2 \quad 1 \le i \le l, \tag{107}$$

$$\widetilde{\mathcal{E}}_{T_i}(\theta_i, \mathcal{S}_i)^2 = \mathcal{E}_T^{int_i}(\theta_i, \mathcal{S}_{int_i})^2 + \mathcal{E}_T^{sb_i}(\theta_i, \mathcal{S}_{sb_i})^2 + \mathcal{E}_T^{tb_i}(\theta_i, \mathcal{S}_{tb_i})^2, \tag{108}$$

where $\mathcal{E}_T^{sb_i}(\theta_i, \mathcal{S}_{sb_i})^2 = \sum_{n=1}^{N_{sb_i}} \omega_{sb_i}^n |R_{sb_i}[u_{\theta_i}](a, t_{sb_i}^n)|^2$, and the remaining terms are defined according to equation (13). Note that $\mathcal{E}_{T_{i-1}}(\theta_{i-1}, \mathcal{S}_{i-1}) = 0$ for $i = 1$.

9.4.3. Error analysis

Let $\hat{u}_i = u_{\theta_i} - u$ denote the error between the HLConcPINN approximation u_{θ_i} and the exact solution u. By applying the convection equation (103) and the definition of the different residuals, we obtain for $1 \le i \le l$,

$$R_{int_i} = \frac{\partial \hat{u}_i}{\partial t} + \frac{\partial \hat{u}_i}{\partial x}, \tag{109a}$$

$$R_{tb_i}|_{t=t_{j-1}} = \hat{u}_i|_{t=t_{j-1}} - \hat{u}_{j-1}|_{t=t_{j-1}} \qquad j = 1, 2, \cdots, i, \tag{109b}$$

$$R_{sb_i}(a,t) = \hat{u}_i(a,t), \tag{109c}$$

where $\hat{u}_0|_{t=t_0} = 0$. We define the total error of the HLConcPINN approximation as $(1 \le i \le l)$

$$\mathcal{E}(\theta_i)^2 = \int_{t_{i-1}}^{t_i} \int_D |\hat{u}_i(\boldsymbol{x}, t)|^2 \, \mathrm{d}x \, \mathrm{d}t. \tag{110}$$

Theorem 9.13. *Let $\widetilde{\Omega}_i = D \times [0, t_i]$. Suppose n, d, $k \in \mathbb{N}$ with $n \ge 2$ and $k \ge 3$, and $u \in H^k(\widetilde{\Omega}_i)$. For every integer $N > 5$, there exists a HLConcPINN u_{θ_i} such that*

$$\|R_{int_i}\|_{L^2(\widetilde{\Omega}_i)}, \ \|R_{sb_i}\|_{L^2(\{a\}\times[0,t_i])}, \ \|R_{tb_i}(x, t_{j-1})\|_{L^2(D)} \lesssim N^{-k+1} \ln N \qquad 1 \le j \le i. \tag{111}$$

Proof. The proof follows from $u \in H^k(\widetilde{\Omega}_i)$, Lemmas 9.2 and 9.8. □

Theorem 9.14. *Let $u \in C^1(\widetilde{\Omega}_i)$ be the classical solution to (103). Let u_{θ_i} $(1 \le i \le l)$ be a HLConcPINN with parameter θ_i. Then the following relation holds,*

$$\int_a^b |\hat{u}_i(x,\tau)|^2 \, \mathrm{d}x \le C_{G_i} \exp(\Delta t) \quad \tau \in [t_{i-1}, t_i], \qquad \int_{t_{i-1}}^{t_i} \int_a^b |\hat{u}_i(x,t)|^2 \, \mathrm{d}x \, \mathrm{d}t \le C_{G_i} \Delta t \exp(\Delta t), \tag{112}$$

where

$$\begin{aligned} &C_{G_i} = 2C_{G_{i-1}} \exp(\Delta t) + \widetilde{C}_{G_i}, \qquad C_{G_0} = 0, \\ &\widetilde{C}_{G_i} = 2 \sum_{j=1}^{i} \int_a^b |R_{tb_i}(x, t_{j-1})|^2 dx + \int_{t_{i-1}}^{t_i} \int_a^b |R_{int_i}|^2 dx dt + \int_{t_{i-1}}^{t_i} |R_{sb_i}|^2 dt. \end{aligned} \tag{113}$$

Proof. The result (112) can be attained by following the same strategy as in the proofs of Theorems 3.4 and 4.2. □

Theorem 9.15. *Let $u \in C^3(\widetilde{\Omega}_i)$ be the classical solution of the convection equation (103), and let u_{θ_i} $(1 \le i \le l)$ be a HLConcPINN with parameter θ_i. Then the total approximation error satisfies*

$$\begin{aligned} &\int_{t_{i-1}}^{t_i} \int_a^b |\hat{u}_i(x,t)|^2 \, \mathrm{d}x \, \mathrm{d}t \le C_{T_i} \Delta t \exp(\Delta t) \\ &\quad = \mathcal{O}(\mathcal{E}_{T_i}(\theta_i, \mathcal{S}_i)^2 + M_{int_i}^{-\frac{2}{d+1}} + M_{tb_i}^{-\frac{2}{d}} + M_{sb_i}^{-\frac{2}{d}}), \end{aligned} \tag{114}$$

where

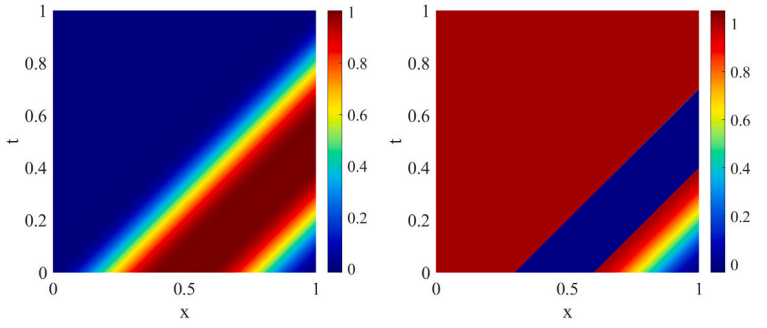


Fig. 19. Convection equation: Distributions of the HLConcPINN solution u with (a) the continuous initial distribution, and (b) the discontinuous (square-wave) initial distribution. NN: [2,90,90,10,1], with tanh activation function for all hidden layers.

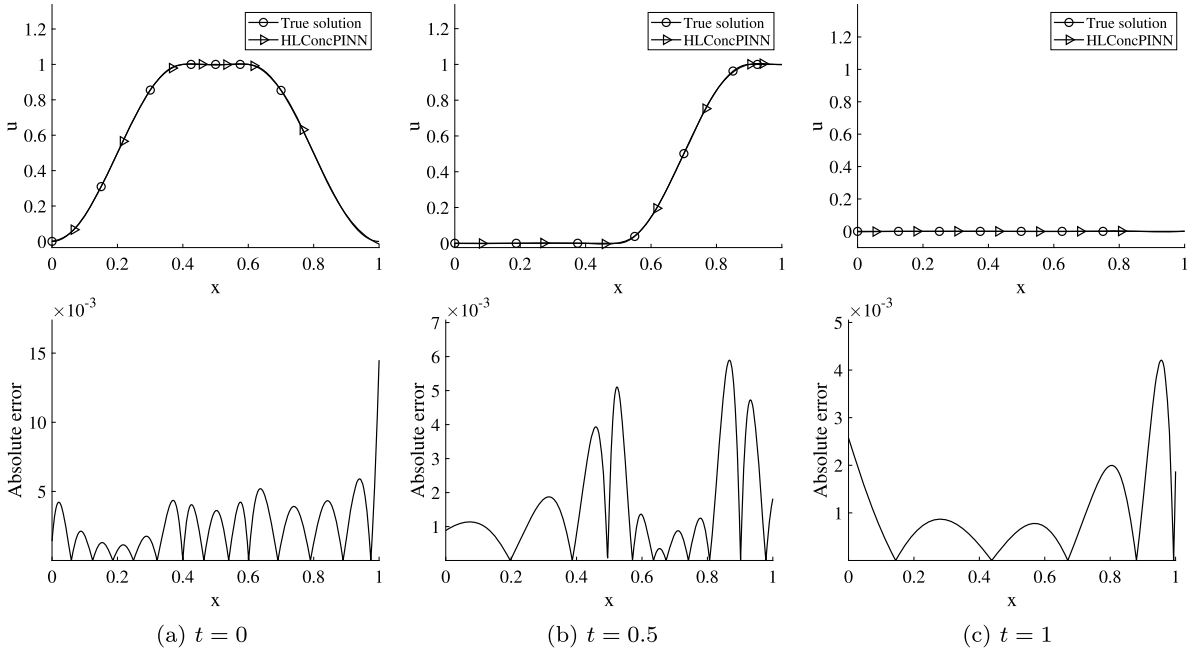


Fig. 20. Convection equation (case I): Top row, comparison of profiles between the true solution and the HLConcPINN solution at (a) $t = 0$, (b) $t = 0.5$, and (c) $t = 1$. Bottom row, absolute-error profiles of the HLConcPINN solution for u. Simulation parameters follow those of Fig. 19.

$$C_{T_i} = 2C_{T_{i-1}} \exp(\Delta t) + \widetilde{C}_{T_i}, \qquad C_{T_0} = 0, \tag{115}$$

$$\widetilde{C}_{T_i} = 2\sum_{j=1}^{i} \left(C_{(R^2_{tb_i}(x,t_{j-1}))} M_{tb_i}^{-\frac{2}{d}} + \mathcal{Q}^{D}_{M_{tb_i}}[R^2_{tb_i}(x,t_{j-1})] \right) + C_{(R^2_{int_i})} M_{int_i}^{-\frac{2}{d+1}} + \mathcal{Q}^{\Omega_i}_{M_{int_i}}[R^2_{int_i}]$$
$$+ C_{(R^2_{sb_i})} M_{sb_i}^{-\frac{2}{d}} + \mathcal{Q}^{\Omega_{si}}_{M_{sb_i}}[R^2_{sb_i}]. \tag{116}$$

The symbols $\mathcal{Q}^{D}_{M_{tb_i}}[R^2_{tb_i}]$, $\mathcal{Q}^{\Omega_i}_{M_{int_i}}[R^2_{int_i}]$ and $\mathcal{Q}^{\Omega_{si}}_{M_{sb_i}}[R^2_{sb_i}]$ denote the midpoint rule, as described in (3).

Proof. The proof follows from Lemma 9.3, Theorem 9.14, and the quadrature error formula (3). □

9.4.4. Numerical examples

We apply the HLConcPINN-ExBTM method to simulate the convection problem (103a)– (103c) with a continuous and a discontinuous initial distribution. Consider a space-time domain $D \times [0,T] = [0,1] \times [0,1]$, and we first employ one time block in the simulations. An architecture $[2, 90, 90, 10, 1]$ has been used for HLConcPINN-ExBTM with the tanh activation function for all hidden layers. In the numerical experiments, we distribute 3000 uniform random collocation points in the interior of domain and on the

boundary $x = 0$. The collocation points on the initial boundary ($t = 0$) are distributed differently depending on the initial distribution, as follows:

- Case I: The continuous initial distribution $u_{in}(x)$ given by,

$$u_{in}(x) = \begin{cases} \cos\left(\frac{5\pi}{2}x - \pi\right), & 0 \le x < 0.4, \\ 1, & 0.4 \le x < 0.6, \\ \cos\left(\frac{5\pi}{2}\pi x - \frac{3\pi}{2}\right), & 0.6 \le x \le 1, \end{cases}$$

  with the boundary condition $g(t) := u(0,t) = 0$. The top plot of Fig. 20(a) visualizes this distribution. For this case, we employ 600 uniform grid points on each sub-interval (i.e. $0 \le x < 0.4$, $0.4 \le x < 0.6$ and $0.6 \le x \le 1$) of the initial boundary.
- Case II: The discontinuous (square wave) initial distribution $u_{in}(x)$ given by

$$u_{in}(x) = \begin{cases} 1, & 0 \le x < 0.3, \\ 0, & 0.3 \le x < 0.6, \\ \cos\left(\frac{5\pi}{2}\pi x - \frac{3\pi}{2}\right), & 0.6 \le x \le 1, \end{cases}$$

  with the boundary condition $g(t) := u(0,t) = 1$. The top plot of Fig. 21(a) illustrates this initial distribution. We similarly employ 600 uniform grid points on each sub-interval (i.e. $0 \le x < 0.3$, $0.3 \le x < 0.6$ and $0.6 \le x \le 1$).

With the above settings, the training loss function in (105)– (106) reduces to the following form:

$$Loss = \frac{W_1}{N_{c1}} \sum_{n=1}^{N_{c1}} \left[\frac{\partial u_\theta}{\partial t}(x_{int}^n, t_{int}^n) + \frac{\partial u_\theta}{\partial x}(x_{int}^n, t_{int}^n) \right]^2 + \frac{W_2}{N_{c2}} \sum_{n=1}^{N_{c2}} \left[u_\theta(x_{tb}^n, 0) - u_{in}(x_{tb}^n) \right]^2 \tag{117}$$

$$+ \frac{W_3}{N_{c3}} \sum_{n=1}^{N_{c3}} \left[u_\theta(0, t_{sb}^n) - g(t_{sb}^n) \right]^2, \tag{118}$$

where the penalty coefficients are set as $(W_1, W_2, W_3) = (0.1, 0.9, 0.9)$ and $(N_{c1}, N_{c2}, N_{c3}) = (3000, 3000, 1800)$. We train the neural network with the Adam optimizer for 1000 epochs, followed by the L-BFGS optimizer for 4000 iterations.

Fig. 19 depicts the distributions of the solution field in the space-time domain corresponding to the continuous (case I) and discontinuous (case II) initial distributions, showing the transport of the initial bump (case I) or the square well (case II) rightward over time and the eventual exit from the domain. Figs. 20 and 21 illustrate profiles of the HLConcPINN-ExBTM solution u at several time instants ($t = 0, 0.5, 1$ in Fig. 20, and $t = 0, 0.1, 0.2, 0.3, 0.5, 1$ in Fig. 21), along with their absolute errors, corresponding to these two cases. The profiles of the true solution for both cases are included for comparison. The numerical results indicate that, for the continuous initial distribution (Case I) the HLConcPINN solution is quite accurate. For case II with the discontinuous initial distribution, the HLConcPINN solution is less accurate compared with the first case. Large errors are induced near the points of physical discontinuity. However, away from the physical discontinuities, the HLConcPINN solution is in good agreement with the physical solution. Overall, the current method has captured the characteristics of the discontinuous physical solution of this problem reasonably well. On the other hand, the large errors at the discontinuity also highlight the challenges facing the method for this type of problems.

We next focus on the discontinuous initial distribution (case II) and consider the use of multiple time blocks in HLConcPINN-ExBTM for simulating this problem. The training loss function in (105)– (106) for HLConcPINN-ExBTM on time block i ($1 \le i \le l$, l denoting the number of time blocks) then becomes,

$$\begin{aligned} Loss_i = & \frac{W_1}{N_{c1}} \sum_{n=1}^{N_{c1}} \left[\frac{\partial u_{\theta_i}}{\partial t}(x_{int}^n, t_{int}^n) + \frac{\partial u_{\theta_i}}{\partial x}(x_{int}^n, t_{int}^n) \right]^2 + \frac{W_2}{N_{c2}} \sum_{n=1}^{N_{c2}} \left[u_{\theta_i}(x_{tb}^n, 0) - u_{in}(x_{tb}^n) \right]^2 \\ & + \frac{W_3}{N_{c3}} \sum_{n=1}^{N_{c3}} \left[u_{\theta_i}(0, t_{sb}^n) - g(t_{sb}^n) \right]^2 + \frac{W_4}{N_{c4}} \sum_{j=1}^{i} \sum_{n=1}^{N_{c4}} \left[u_{\theta_i}(x_{tb}^n, t_{j-1}) - u_{\theta_{j-1}}(x_{tb}^n, t_{j-1}) \right]^2 \\ & + Loss_{i-1}, \end{aligned}$$

where $Loss_0 = 0$, and $u_{\theta_0}(x, t_0) = u_{in}(x)$. Here we employ the same W_k, N_{ck} ($k = 1, 2, 3$) values as in (117) (for the case of one time block), together with $W_4 = 0.9$ and $N_{c4} = 3000$. In each time block, the training process is similar to that with a single time block, with the same parameters (e.g. number of epochs, optimizers) as described above.

Fig. 22 shows distributions of the HLConcPINN-ExBTM solutions obtained using two uniform time blocks and four uniform time blocks in the space-time domain, respectively. Fig. 23 compares the u profiles between the true solution and the HLConcPINN-ExBTM solution at time instants $t = 0, 0.2, 0.45$, and 0.8, and shows the corresponding absolute-error profiles of the HLConcPINN solution.

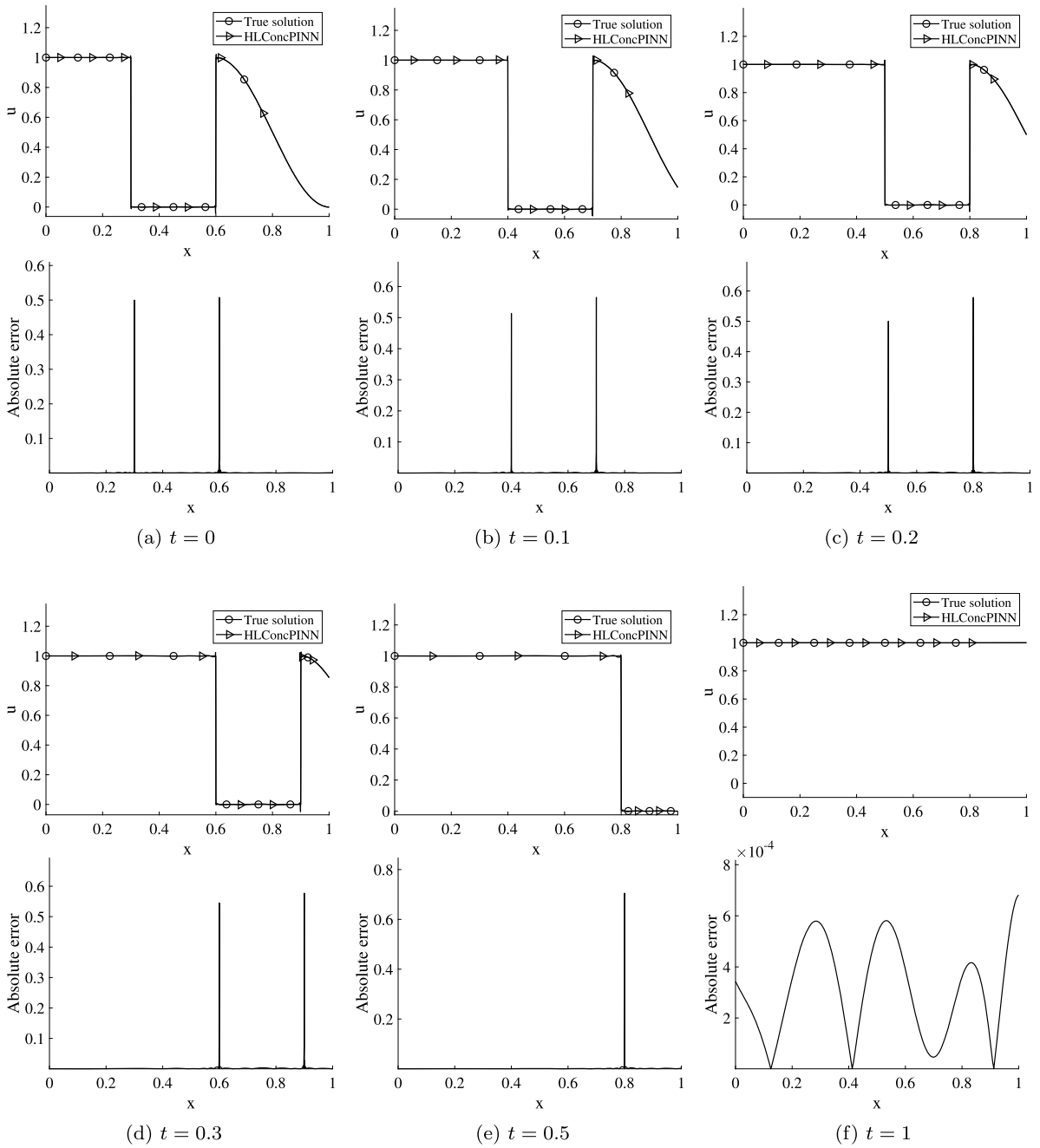
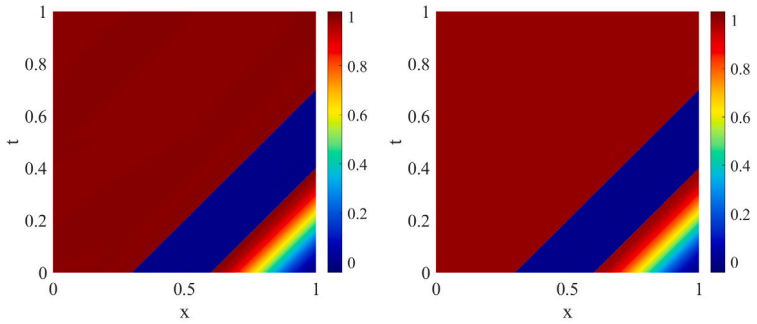


Fig. 21. Convection equation (case II): In each sub-figure, top row: comparison of profiles between the true solution and the HLConcPINN solution at (a) $t = 0$, (b) $t = 0.1$, (c) $t = 0.2$, (d) $t = 0.3$, (e) $t = 0.5$, and (f) $t = 1$; bottom row: absolute-error profiles of the HLConcPINN solution for u. Simulation parameters follow those of Fig. 19 (a single time block in domain).

It is evident that the solutions obtained using multiple time blocks are very close to that obtained with a single time block in the domain.

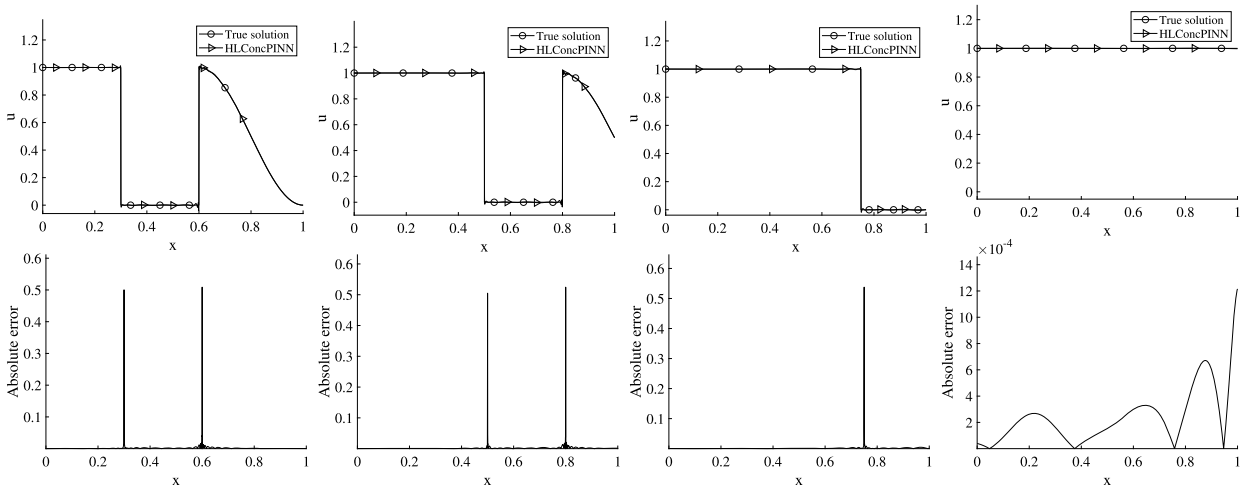
CRediT authorship contribution statement

Yanxia Qian: Writing – review & editing, Writing – original draft, Investigation, Funding acquisition, Formal analysis, Data curation. **Yongchao Zhang:** Writing – review & editing, Writing – original draft, Visualization, Software, Funding acquisition, Data curation. **Suchuan Dong:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Funding acquisition, Formal analysis, Conceptualization.

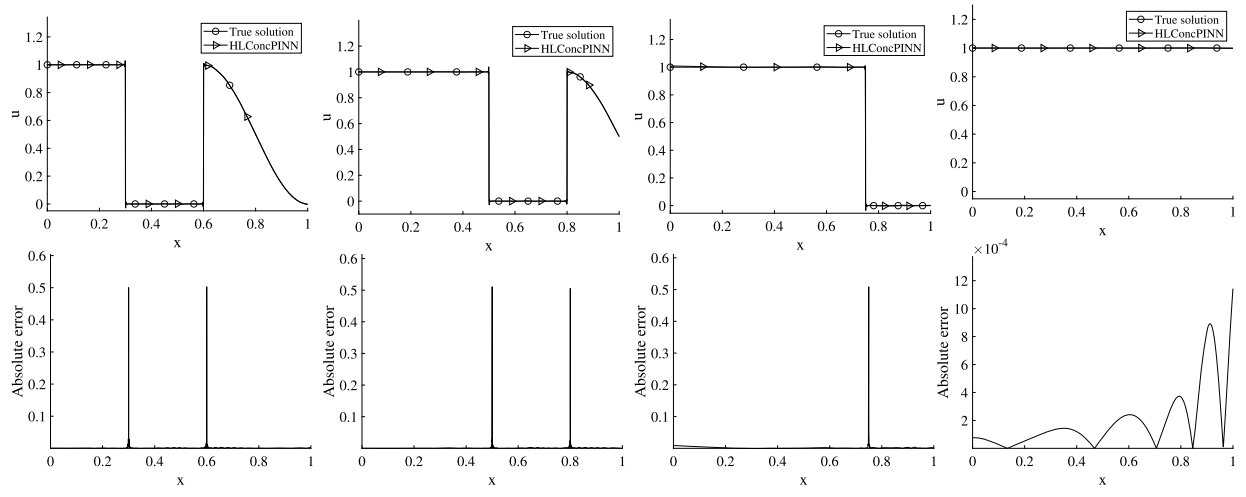


(a) Two uniform time blocks (b) Four uniform time blocks

Fig. 22. Convection equation (case II): Distributions of the HLConcPINN solution for u obtained with (a) two and (b) four uniform time blocks in the space-time domain. NN: [2,90,90,10,1], with tanh activation function for all hidden layers.



(a) Two time blocks: $t = 0$ (b) Two time blocks: $t = 0.2$ (c) Two time blocks: $t = 0.45$ (d) Two time blocks: $t = 0.8$



(e) Four time blocks: $t = 0$ (f) Four time blocks: $t = 0.2$ (g) Four time blocks: $t = 0.45$ (h) Four time blocks: $t = 0.8$

Fig. 23. Convection equation (case II): In each sub-figure, top plot: comparison of profiles between the true solution and the HLConcPINN solution for u at (a,e) $t = 0$, (b,f) $t = 0.2$, (c,g) $t = 0.45$, (d,h) $t = 0.8$; bottom plot: absolute-error profiles of the HLConcPINN solution for u. Plots (a)-(d) are obtained with two time blocks and (e)-(h) are obtained with four time blocks in HLConcPINN.

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.

Data availability

Data will be made available on request.

References

[1] G. Bai, U. Koley, S. Mishra, R. Molinaro, Physics informed neural networks (PINNs) for approximating nonlinear dispersive PDEs, J. Comput. Math. 39 (6) (2021) 816–847.
[2] I.A. Baratta, J.P. Dean, J.S. Dokken, M. Habera, J.S. Hale, C.N. Richardson, M.E. Rognes, M.W. Scroggs, N. Sime, G.N. Wells, DOLFINx: the next generation FEniCS problem solving environment, 2023.
[3] C. Beck, W. E, A. Jentzen, Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations, J. Nonlinear Sci. 29 (4) (2019) 1563–1619.
[4] J. Berner, P. Grohs, A. Jentzen, Analysis of the generalization error: empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations, SIAM J. Math. Data Sci. 2 (3) (2020) 631–657.
[5] S. Berrone, C. Canuto, M. Pintore, Solving PDEs by variational physics-informed neural networks: an a posteriori error analysis, Ann. Univ. Ferrara, Sez. 7: Sci. Mat. 68 (2) (2022) 575–595.
[6] A. Biswas, J. Tian, S. Ulusoy, Error estimates for deep learning methods in fluid dynamics, Numer. Math. 151 (3) (2022) 753–777.
[7] Z. Cai, J. Chen, M. Liu, X. Liu, Deep least-squares methods: an unsupervised learning-based numerical method for solving elliptic PDEs, J. Comput. Phys. 420 (2020) 109707.
[8] F. Calabro, G. Fabiani, C. Siettos, Extreme learning machine collocation for the numerical solution of elliptic PDEs with sharp gradients, Comput. Methods Appl. Mech. Eng. 387 (2021) 114188.
[9] S. Cuomo, V. Schiano Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-informed neural networks: where we are and what's next, J. Sci. Comput. 92 (3) (2022) 88.
[10] E. Cyr, M. Gulian, R. Patel, M. Perego, N. Trask, Robust training and initialization of deep neural networks: an adaptive basis viewpoint, Proc. Mach. Learn. Res. 107 (2020) 512–536.
[11] P. Davis, P. Rabinowitz, Methods of Numerical Integration, Dover Publications, Inc, 2007.
[12] T. De Ryck, A.D. Jagtap, S. Mishra, Error estimates for physics-informed neural networks approximating the Navier–Stokes equations, IMA J. Numer. Anal. 44 (1) (2023) 83–119.
[13] T. De Ryck, S. Lanthaler, S. Mishra, On the approximation of functions by tanh neural networks, Neural Netw. 143 (2021) 732–750.
[14] S. Dong, Z. Li, Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations, Comput. Methods Appl. Mech. Eng. 387 (2021) 114129.
[15] S. Dong, Z. Li, A modified batch intrinsic plasticity method for pre-training the random coefficients of extreme learning machines, J. Comput. Phys. 445 (2021) 110585.
[16] S. Dong, N. Ni, A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks, J. Comput. Phys. 435 (2021) 110242.
[17] S. Dong, Y. Wang, A method for computing inverse parametric PDE problems with random-weight neural networks, J. Comput. Phys. 489 (2023) 112263, also arXiv:2210.04338.
[18] S. Dong, J. Yang, Numerical approximation of partial differential equations by a variable projection method with artificial neural networks, Comput. Methods Appl. Mech. Eng. 398 (2022) 115284, also arXiv:2201.09989.
[19] S. Dong, J. Yang, On computing the hyperparameter of extreme learning machines: algorithms and applications to computational PDEs, and comparison with classical and high-order finite elements, J. Comput. Phys. 463 (2022) 111290, also, arXiv:2110.14121.
[20] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (2018) 1–12.
[21] L.C. Evans, Partial Differential Equations, second edn., American Mathematical Society, Providence, RI, 2010.
[22] G. Fabiani, F. Calabro, L. Russo, C. Siettos, Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines, J. Sci. Comput. 89 (2021) 44.
[23] G. Fabiani, E. Galaris, L. Russo, C. Siettos, Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs, Chaos 33 (4) (2023) 043128.
[24] J. Gao, Y. Zakharian, PINNs error estimates for nonlinear equations in R-smooth Banach spaces, arXiv:2305.11915.
[25] J. He, J. Xu, MgNet: a unified framework for multigrid and convolutional neural network, Sci. China Math. 62 (2019) 1331–1354.
[26] R. Hu, Q. Lin, A. Raydan, S. Tang, Higher-order error estimates for physics-informed neural networks approximating the primitive equations, Part. Differ. Equ. Appl. 4 (4) (2023) 34.
[27] Z. Hu, A.D. Jagtap, G.E. Karniadakis, K. Kawaguchi, When do extended physics-informed neural networks (XPINNs) improve generalization?, SIAM J. Sci. Comput. 44 (5) (2022) A3158–A3182.
[28] Z. Hu, C. Liu, Y. Wang, Z. Xu, Energetic variational neural network discretizations to gradient flows, arXiv:2206.07303.
[29] A. Jagtap, G. Karniadakis, Extended physics-informed neural network (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (2020) 2002–2041.
[30] A. Jagtap, E. Kharazmi, G. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, Comput. Methods Appl. Mech. Eng. 365 (2020) 113028.
[31] G. Karniadakis, G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (2021) 422–440.
[32] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 26548–26560.
[33] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[34] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: a deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.
[35] S. Mishra, R. Molinaro, Physics informed neural networks for simulating radiative transfer, J. Quant. Spectrosc. Radiat. Transf. 270 (2021) 107705.
[36] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs, IMA J. Numer. Anal. 42 (2) (2022) 981–1022.

[37] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating PDEs, IMA J. Numer. Anal. 43 (1) (2023) 1–43.
[38] N. Ni, S. Dong, Numerical computation of partial differential equations by hidden-layer concatenated extreme learning machine, J. Sci. Comput. 95 (2) (2023) 35.
[39] P. Pantidis, H. Eldababy, C.M. Tagle, M.E. Mobasher, Error convergence and engineering-guided hyperparameter search of PINNs: towards optimized I-FENN performance, Comput. Methods Appl. Mech. Eng. 414 (2023) 116160.
[40] M. Penwarden, A.D. Jagtap, S. Zhe, G.E. Karniadakis, R.M. Kirby, A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions, J. Comput. Phys. 493 (2023) 112464.
[41] Y. Qian, Y. Zhang, Y. Huang, S. Dong, Physics-informed neural networks for approximating dynamic (hyperbolic) PDEs of second order in time: error analysis and numerical algorithms, J. Comput. Phys. 495 (2023) 112527.
[42] M. Raissi, P. Perdikaris, 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. 378 (2019) 686–707.
[43] Y. Shin, J. Darbon, G.E. Karniadakis, On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs, Commun. Comput. Phys. 28 (5) (2020) 2042–2074.
[44] Y. Shin, Z. Zhang, G.E. Karniadakis, Error estimates of residual minimization using neural networks for linear PDEs, J. Mech. Learn. Model. Comput. 4 (4) (2023) 73–101.
[45] J.W. Siegel, Q. Hong, X. Jin, W. Hao, J. Xu, Greedy training algorithms for neural networks and applications to PDEs, J. Comput. Phys. 484 (112084) (2023) 27.
[46] J. Sirignano, K. Spoliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[47] A. Tartakovsky, C. Marrero, P. Perdikaris, G. Tartakovsky, D. Barajas-Solano, Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems, Water Resour. Res. 56 (2020) e2019WR026731.
[48] R. Temam, Infinite-Dimensional Dynamical Systems in Mechanics and Physics, second edn., Springer-Verlag, New York, 1997.
[49] X. Wan, S. Wei, VAE-KRnet and its applications to variational Bayes, Commun. Comput. Phys. 31 (2022) 1049–1082.
[50] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: a neural tangent kernel perspective, J. Comput. Phys. 449 (2022) 110768.
[51] Y. Wang, S. Dong, An extreme learning machine-based method for computational PDEs in higher dimensions, Comput. Methods Appl. Mech. Eng. 418 (2024) 116578.
[52] Y. Wang, G. Lin, Efficient deep learning techniques for multiphase flow simulation in heterogeneous porous media, J. Comput. Phys. 401 (2020) 108968.
[53] U. Zerbinati, PINNs and GaLS: a priori error estimates for shallow physics informed neural networks applied to elliptic problems, IFAC-PapersOnLine 55 (20) (2022) 61–66.