Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 406 (2023) 115902
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Physics-integrated neural differentiable (PiNDiff) model for composites manufacturing

Deepak Akhare[a], Tengfei Luo[a,b,c], Jian-Xun Wang[a,b,d,*]

[a] *Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN, USA*
[b] *Center for Sustainable Energy (ND Energy), University of Notre Dame, Notre Dame, IN, USA*
[c] *Department of Chemical and Biomolecular Engineering, University of Notre Dame, Notre Dame, IN, USA*
[d] *Lucy Family Institute for Data & Society, University of Notre Dame, Notre Dame, IN, USA*

Received 19 August 2022; received in revised form 21 December 2022; accepted 12 January 2023
Available online 19 January 2023

Abstract

Various manufacturing technologies are being developed to improve the manufacturing of composites owing to their low weight and high performance. The mechanical properties of the composites depend on various variables and parameters of the manufacturing process, which are challenging, if not impossible, to determine and optimize experimentally. Traditional first-principle modeling approaches are not accessible due to the complex physics involved. A hybrid model that combines incomplete physics knowledge with available measurement data within a differentiable programming framework opens up new avenues to tackle the challenges. In this work, a physics-integrated neural differentiable (PiNDiff) model is developed, where the partially known physics is integrated into the recurrent network architecture to enable effective learning and generalization. The merit and potential of the proposed method have been demonstrated in modeling the curing process of thick thermoset composite laminates, whose governing physics is partially given. The proposed PiNDiff model shows the capability to learn unknown physics from the limited, indirect data and, meanwhile, can be used to infer unobserved variables and parameters. The performance of the PiNDiff model has been compared with two state-of-the-art (SOTA) black-box deep learning models, and its advantages over the purely data-driven models and first-principles physics-based models have been discussed in detail. The demonstrated PiNDiff strategy may provide a general strategy to model phenomena where physics is only partially known and sparse, indirect data are available.

Keywords: Composite cure; Differentiable programming; Neural networks; Scientific machine learning; Surrogate modeling; Operator learning

1. Introduction

Composites are a class of materials actively considered for various applications in today's aerospace, automotive, and civil industries owing to their high strength and lightweight. The performance, quality, and repeatability of the composite materials are greatly influenced by the manufacturing process. Great effort has been made in optimizing the manufacturing processes to fabricate new composites with desired properties [1]. For example, aerospace industries are actively improving manufacturing techniques to produce new carbon fiber-reinforced carbon

* Corresponding author at: Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN, USA.
E-mail address: jwang33@nd.edu (J.-X. Wang).

https://doi.org/10.1016/j.cma.2023.115902
0045-7825/

(C/C) composites that can work in harsh operational environments with extremely high temperatures, stress, and strong erosion [2]. However, optimizing composite manufacturing processes manually to meet certain specified requirements is very challenging since most state-of-the-art (SOTA) processing techniques for composites are sophisticated and involve many inter-dependent steps, which are usually very time-consuming [3–5]. For instance, the manufacturing of C/C composites involves impregnation, carbonization (pyrolysis), and densification processes, which are conducted iteratively and can take several months to reach the final state [6], making trial-and-error improvements intractable.

Therefore, computer-based predictive modeling of composite manufacturing is essential for controlling the process, shortening the production cycle, and optimizing material properties to meet the requirements of various applications. Although computer models have been successfully used to predict various physical processes in many applications (e.g., aerospace [7,8], structure [9], biomechanics [10,11], etc.), the physics-based computational modeling for composite manufacturing is a less explored area, and there are only very few studies on modeling the impregnation or pyrolysis processes in past decades [12–19]. This is because the manufacturing processes of composites involve many coupled physics and complex mechanisms, both mechanically and chemically, which are far from being fully understood, making traditional first-principles physics-based modeling infeasible.

Recent advances in machine learning (ML) and deep neural networks (DNN), combined with the ever-increasing data availability, opened up new avenues for developing predictive modeling of composite manufacturing based on massive amounts of data [20–22]. In the past few years, data-driven modeling has emerged as a practical approach in additive manufacturing, allowing for the automatic discovery of patterns and trends in data, construction of quantitative models of process-structure–property relationships over the parameter space, and prediction at unseen points [20]. Moreover, ML techniques have been used to discover complex constitutive laws of composites from data [21], and to build fast surrogate DNN models, facilitating multi-scale predictive modeling of composite materials [23]. For example, Huang et al. [22] developed a data-driven model to predict the mechanical properties of carbon nanotube (CNT)-reinforced cement composites, demonstrating better generalizability and predictability than traditional response surface-based methods. Nguyen et al. [24] proposed a neural network-based constitutive model to capture the evolution of the matrix mechanical properties as a function of temperature and degree of cure in the composites manufacturing process, and Kopal et al. [25] utilized neural networks to predict the curing characteristics of carbon black-filled rubber blends. Tao et al. [26] used experimental data to discover failure criteria of composites within a sparse regression framework, which promotes sparsity to find the most parsimonious model form. Baek et al. [27] employed DNN to establish a structure–property relationship of polymer nanocomposites.

Although showing great promise, purely data-based DNN models heavily rely on “big data” and often suffer from the generalizability issue in out-of-training regimes, impeding their effective applications in modeling complex composite manufacturing processes [21]. To tackle these challenges, physics prior knowledge may be leveraged to inform the neural network design, training and inference, and this strategy is known as physics-informed deep learning (PIDL), which has been explored and demonstrated promising in many fields, including solid mechanics [28–30], turbulent flows [31–33], materials [34], heat transfer [35–38], and biomedical problems [39–41]. One of the important contributions of PIDL is the physics-informed neural network (PINN) [36,42–46], where the loss function is constructed by the residual of governing equations, and thus the training process is constrained by the governing physics, reducing the requirement of labeled data [47]. In particular, PINN has been successfully applied to simulate the thermochemical evolution of composite materials based on the coupled PDEs that describe heat conduction and resin cure kinetics [36]. In general, the physics utilized in PINNs is in the form of complete governing equations and imposed as (soft) constraints during the training process. The nonlinearity introduced into the loss function would pose significant challenges in optimization, making DNN training difficult [48]. Moreover, the complete analytical forms of the governing physics are assumed to be known in PINNs, and it is not straightforward to leverage incomplete physics, where the governing equations are only partially known and model forms are incomplete.

For composite manufacturing, the physics involved has not been fully understood due to the high complexity, and available sensing data collected during the manufacturing process is often sparse and indirect to the quantity of interest (QoI), making most existing purely physics-based, data-driven, or PIDL models inapplicable. Therefore, it is imperative to develop a physics-informed, data-enabled deep learning framework that can leverage indirect sensing data and available physics prior, which is incomplete or imperfect. To this end, we propose a novel *physics-integrated neural differentiable (PiNDiff) model*, a physics-informed deep learning framework for composite manufacturing

that integrates partially-known physics of the manufacturing process into the DNN architecture, preserving the mathematical structure of incomplete governing equations using differentiable programming (∂P) [49]. The general idea is to represent the (partially)-known physics as a fully differentiable numerical model, integrated into the neural networks as a joint deep learning (DL) architecture. The automatic differentiation technique enables the gradient information to back-propagate throughout the entire computer program, and thus the hybrid neural model can be trained in the same way as the classic DNNs. By integrating physics-based equations into DL models, the full potential of each element can be well utilized, significantly enhancing learning capability from limited data. Very recently, this idea has been explored and demonstrated effective in several different fields [49–52]. For example, Liu et al. [53] developed a PDE-preserved neural network (PPNN) for surrogate modeling of spatiotemporal physics by preserving partially known governing PDEs as multi-resolution convolutional residual network blocks. Huang et al. [54] embedded a neural network into a finite element solver to learn a constitutive relation of nonlinear materials from indirect data. These studies imply a great promise of integrating differentiable physics into DNN architectures.

In this work, we introduce the proposed PiNDiff model and demonstrate its merit by comparing it with two SOTA pure data-driven DL models. As an important step in many composites (e.g., polymer matrix and C/C composites), the curing process will be studied to showcase the proposed method, where the thermochemical evolution physics of the composite can be rapidly predicted given different boundary conditions(BC). The performance of the proposed model is investigated and compared with existing SOTA DL models in terms of predictive accuracy, generalizability, robustness, and sample efficiency. The demonstrated PiNDiff model can generally be applicable to other problems where physics is only partially known, and data are limited. The rest of the paper is organized as follows: the overall methodology of the proposed PiNDiff model and curing process physics are introduced in Section 2. Results of numerical experiments of PiNDiff and its comparison with SOTA DL models are presented in Section 3. Several features of the proposed method are discussed in Section 4. Finally, Section 5 concludes the paper.

2. Methodology

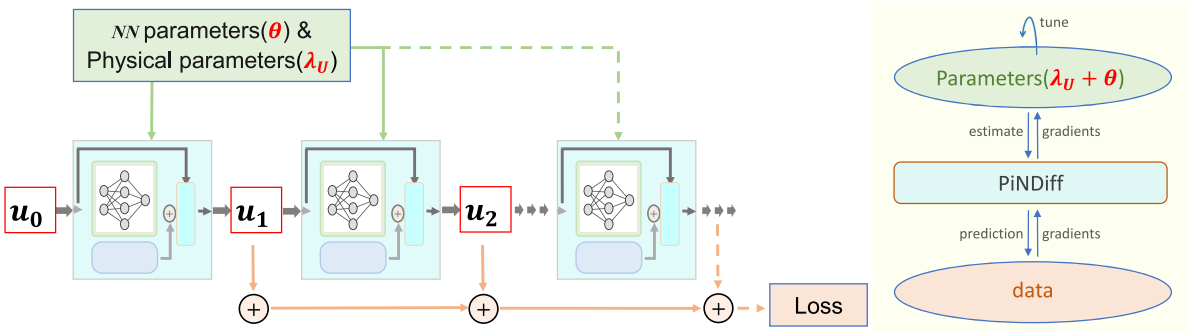
2.1. Problem formulation

Manufacturing polymer matrix composites or C/C composites usually involves a combination of different processes such as impregnation of resin into the fiber preform, carbonization of polymers, chemical vapor infiltration of carbon using gaseous hydrocarbon precursors, and graphitization of the composite materials. These processes involve fluid flows, gas diffusion, heat transfer, polymerization, and deposition reactions, which can be described by a system of coupled partial differential equations (PDEs) (Eq. (1a)) with boundary condition (Eq. (1b)),

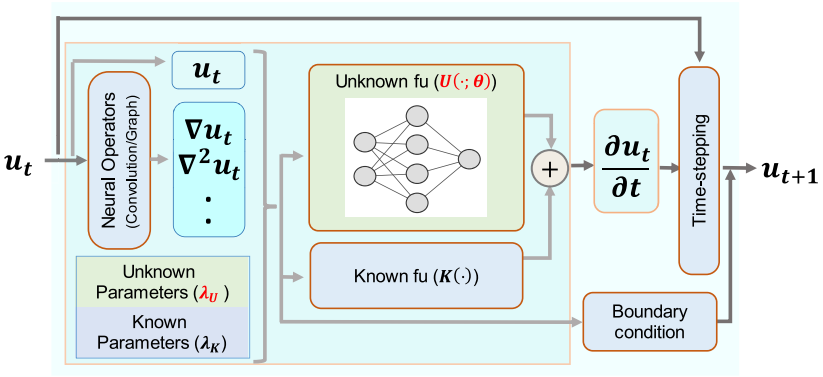
$$\frac{\partial \mathbf{u}(\mathbf{x})}{\partial t} = \mathcal{K}(\mathbf{u}(\mathbf{x}), \nabla\mathbf{u}, \nabla^2\mathbf{u}..., \boldsymbol{\lambda}_K, \boldsymbol{\lambda}_U) + \mathcal{U}(\mathbf{u}(\mathbf{x}), \nabla\mathbf{u}, \nabla^2\mathbf{u}..., \boldsymbol{\lambda}_K, \boldsymbol{\lambda}_U) \quad \mathbf{x}, t \in \Omega_{p,t}, \tag{1a}$$

$$\mathcal{B}(t, \mathbf{u}(\mathbf{x}), \nabla\mathbf{u}, \boldsymbol{\lambda}_K, \boldsymbol{\lambda}_U) = 0 \quad \text{in} \quad \mathbf{x}, t \in \partial\Omega_{p,t}, \tag{1b}$$

where $\mathbf{x}$, t are space and time coordinates, respectively; $\Omega_{p,t} \triangleq \Omega_p \times [0, T_r]$ with Ω_p representing the physical domain and T_r as the time range; $\mathcal{K}$ and $\mathcal{U}$ respectively represent the known and unknown portions of the nonlinear function of the state vector $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^n$ and its derivatives ($\nabla\mathbf{u}$, $\nabla^2\mathbf{u}$...), defined on the physical domain Ω_p. Both the known $\mathcal{K}(\cdot)$ and unknown $\mathcal{U}(\cdot)$ functions depend on physical parameters, whose values are also sometimes unknown. Here, we use $\boldsymbol{\lambda}_K$ and $\boldsymbol{\lambda}_U$ to represent the known and unknown physical parameters, respectively. The boundary conditions such as heat convection enforced on boundary $\partial\Omega_p$ can be generically represented by a differential operator $\mathcal{B}$ as shown in Eq. (1b), which is also parameter-dependent. In most cases, the functional form $\mathcal{B}(\cdot)$ of the boundary condition is given, and thus this study assumes that there are certain parameters boundary $\boldsymbol{\lambda}_U$ might be unknown. However, the proposed framework is also able to handle boundary conditions with unknown model forms. The goal of this work is to develop a computational framework for predictive modeling of this kind of system, where the physics knowledge is only partially known, and a small amount of indirect observation data can be obtained sparsely.



(a) Overview of the PiNDiff architecture



(b) PiNDiff module of one-step prediction

Fig. 1. (a) The overview of the auto-regressive learning architecture of PiNDiff model and (b) Zoom-in view of the PiNDiff module for one time-step prediction (fu: function).

2.2. *Physics-integrated neural differentiable (PiNDiff) modeling*

The missing physics poses significant challenges in direct physics-based modeling, while the available sensing data is often sparse and indirect to QoIs, making purely data-driven modeling infeasible. To address these challenges, this work presents a novel data-driven framework, PiNDiff, which integrates the partially-known physics into deep learning architecture to effectively model the system with limited, indirect data, enabling fast surrogate prediction and real-time decision-making in the composites manufacturing process. The overview of the proposed PiNDiff framework is shown in Fig. 1(a), where the overall learning architecture is based on recurrent neural networks consisting of a series of physics-integrated convolutional residual connection networks, known as PiNDiff modules (Fig. 1(b)), which are connected in an auto-regressive manner. The sequential convolutional network structure is capable of capturing the spatio-temporal evolution of the physics, where the state variables at each time step are predicted by the PiNDiff module, given the information from previous time steps. The information propagation across the PiNDiff modules is modeled by the NeuralODE [55], which is in contrast to discrete recurrent neural networks with a finite number of hidden layers; the residual connections with infinite depth are described by the continuous ODEs obtained from the governing PDEs using the method of lines. Classic numerical time integration schemes, e.g., Euler or Runge–Kutta methods, can be used for the forward stepping of the neuralODEs, and automatic differentiation (AD) techniques will make these schemes fully differentiable.

Unlike traditional DNN models that are purely black-box, the PiNDiff module is designed to preserve the mathematical structures of the known physics, as shown in Fig. 1(b). Specifically, a PiNDiff module contains several trainable and non-trainable neural networks to represent the partially-known governing PDEs. As both the known

$\mathcal{K}(\cdot)$ and unknown $\mathcal{U}(\cdot)$ portions of the nonlinear functional in Eq. (1a) are composed of spatial derivatives of the solutions, multiple non-trainable neural operations such as convolutional operation [53] (for structured grid) and graph message passing (for the unstructured grid) can be applied to the input state variables to approximate a group of spatial differential terms, e.g., $\nabla(\mathbf{u})$, $\nabla^2(\mathbf{u})$, ..., which will be used to construct the known and unknown functions. These neural operations are defined based on classic numerical schemes, e.g., finite difference stencils, which are kept non-trainable to reduce the network complexity and, thus, largely lower the required labeled data for training. The known portion is directly constructed based on the governing PDEs, while the unknown portion is built by a series of DNNs with trainable parameters $\boldsymbol{\theta}$. Namely, the derivative features are given as inputs to the known functions $\mathcal{K}(\cdot)$ and neural networks functions $\mathcal{U}_{nn}(\cdot)$. The outputs of the known and unknown neural functions are then combined for time stepping to predict the future states. Unknown physical parameters $\boldsymbol{\lambda}_U$ in the governing equations and boundary conditions are also trainable to enable model inference.

In order to train the PiNDiff model as a whole, the entire framework should be fully differentiable. Namely, the gradient information is able to backpropagate throughout the entire framework to enable the use of stochastic gradient descent for the PiNDiff training. This is the general idea of differentiable programming (∂P) [49], the generalization of deep learning. In this work, the automatic differentiation engine in PyTorch [56] is leveraged to power the training of the ∂P model. The trainable parameters $\boldsymbol{\theta}_{tot}$ of the PiNDiff model include both neural networks parameters $\boldsymbol{\theta}$ and unknown physical parameters $\boldsymbol{\lambda}_U$, i.e., $\boldsymbol{\theta}_{tot} = \{\boldsymbol{\theta}, \boldsymbol{\lambda}_U\}$. Labeled data is required to train the PiNDiff model. In contrast to the classic deep learning model, labels $\mathbf{d} \in \mathbb{R}^m$ can be indirect to the state $\mathbf{u} \in \mathbb{R}^n$. This feature can have important practical implications; for example, in a curing process, one can more easily monitor the temperature of the composite as a function of time than its degree of cure, which represents the state of the material. The state-to-observable map $\mathcal{F} : \mathbf{u} \to \mathbf{d}$ is either given or modeled as another neural network, which can be embedded into the ∂P architecture.

The loss function $\mathcal{L}$ is formulated as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}_U) = \sum_{t=0}^{t_n} \|\mathcal{F}\big(\mathbf{u}_t(\boldsymbol{\theta}, \boldsymbol{\lambda}_U)\big) - \hat{\mathbf{d}}_t\|_{L_2} + \beta_1 \|\boldsymbol{\theta}\|_{L_2} + \beta_2 \sum_{t=0}^{t_n-1} \|\mathbf{u}_{t+1}(\boldsymbol{\theta}, \boldsymbol{\lambda}_U) - \mathbf{u}_t(\boldsymbol{\theta}, \boldsymbol{\lambda}_U)\|_{L_2}, \tag{2}$$

where the first term is the L2 norm of the discrepancy between the observable data and the predicted state projected to the observation across all time steps; $\|\cdot\|_{L_2}$ represents the L2 norm; the second and third terms are regularization terms — the former is to promote sparsity, while the latter is to enforce the smoothness of the trajectory; β_1 and β_2 are weighting parameters of regularization terms, which should be small values. The PiNDiff training is conducted by solving the following optimization problem,

$$\boldsymbol{\theta}^*, \boldsymbol{\lambda}_U^* = \underset{\boldsymbol{\theta}, \boldsymbol{\lambda}_U}{\operatorname{argmin}} \ \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}_U). \tag{3}$$

Due to the scale and magnitude difference of neural network parameters θ and unknown physical parameters $\boldsymbol{\lambda}_U$, different initial learning rates, optimizers, and schedulers can be assigned separately. For example, the training will be more efficient by using different learning rates scaled by the orders of magnitude of the parameters to be inferred. If the orders of magnitude $\mathbf{r}$ of the parameters are known *a priori*, which is usually the case, this prior information can be imposed by re-parameterizing the parameter $\boldsymbol{\lambda}_U$ as $\mathbf{v}$, where $\boldsymbol{\lambda}_U = \mathbf{v} \times 10^{\mathbf{r}}$, and then a unified learning rate can be used for the optimization of all parameters.

2.3. Curing process modeling as a showcase

Curing is a common step in the production of polymer matrix composites and C/C composites, as it is the crucial in transforming a loose mass of fibers and resin into engineering composite materials. The process of curing polymer composite laminates involves the impregnation of fibers with a resin and polymerization by applying heat. The curing polymerization is commonly characterized by the dynamic evolution of two physical states in time: the degree of cure α and temperature T. This process involves the coupled physics of heat transfer and thermochemical reactions, which can be described by general conservation laws, but not all the details are fully known. Particularly, the cure kinetics is very complex in real-world scenarios, and exact model forms are not known and have to be approximated rely on many assumptions and rigorous experimental calibration [57]. On the other hand, sparse temperature data can be obtained by placing sensors at certain locations during the process, while other states, such as degree of

cure, are usually not observable on the fly. These features make the curing process modeling an ideal example to showcase the proposed method. Therefore, the PiNDiff method is built for modeling the manufacturing process to cure thick thermoset composite laminates, and the performance is compared against other SOTA black-box deep learning models. To better evaluate and assess the methodology, synthetic data is used in this study, generated from a complete curing model as the ground truth, detailed as follows.

2.3.1. Ground truth model forms

Anandan et al. [58] developed a computational model to describe the cure behavior of a carbon/epoxy prepreg system (IM7/Cycom 5320–1), and the cure kinetic parameters of exothermic reactions are obtained by Differential Scanning Calorimetry. We use this model as the ground truth model form for verification and evaluation of the proposed method. The model involves two fully-coupled differential equations for the cure kinetics and heat transfer inside the composite laminate [57]. The cure kinetics equation accounts for kinetics- and diffusion-controlled reaction mechanisms, which is given as,

$$\frac{d\alpha}{dt} = \sum_{i=1,3} K_i \alpha^{m_i} (1-\alpha)^{n_i} + \sum_{j=2,4} \frac{K_j \alpha^{m_j} (1-\alpha)^{n_j}}{1 + \exp\left(D_j \left(\alpha - \left(\alpha_{C0.j} + \alpha_{CT,j} T\right)\right)\right)}, \tag{4}$$

where α is the degree of cure, accounting for an exothermic reaction, defined as the ratio of heat ΔH_t at time t to the total heat of reaction ΔH_U, i.e., $\alpha = \Delta H_t / \Delta H_U$, T is the temperature, and K_n is Arrhenius temperature dependent term, defined as,

$$K_n = A_n \exp\left(\frac{-E_{A,n}}{RT}\right), n = i, j, \tag{5}$$

where A_n is the Arrhenius constant, $E_{A,n}$ is the activation energy, R is the universal gas constant, D_j is the diffusion constant, m_i and n_i are reaction order-based fitting constants, a_{c0} is the critical degree of cure at absolute zero, and a_{cT} accounts for the increase in the critical degree of cure with temperature.

The energy balance in the laminate portion is described by the heat transfer equation,

$$\rho_c C_c \frac{\partial T}{\partial t} - \nabla \cdot k_c \nabla T = v_r \rho_r H_u \frac{\partial \alpha}{\partial t}, \tag{6}$$

and the heat supplied by the autoclave to the laminate is modeled by a convective heat transfer equation,

$$Q = h(T_{oven}(t) - T_{bc}) = k_c \nabla T \cdot \hat{n} \tag{7}$$

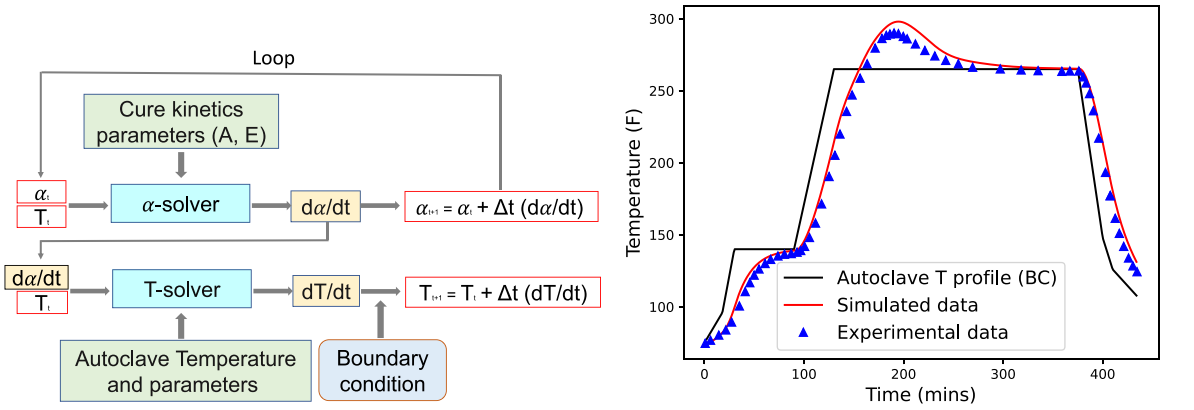
where ρ_c is the density of composite, C_c is the specific heat capacity of composite, k_c is the thermal conductivity of composite, v_r is the resin volume fraction, ρ_r is the resin density, H_u is the ultimate heat of reaction of the system, h is the convective heat transfer coefficient, $\hat{n}$ indicates the normal vector at the boundary surface, T_{oven} and T_{bc} represent the autoclave temperature and the temperature at the boundary surface of the laminate, respectively. The laminate has two thermal energy sources, heat supplied by autoclave and heat generated by exothermic chemical reaction. The term $v_m \rho_m H_u \frac{\partial \alpha}{\partial t}$ on the right-hand side accounts for heat generated by exothermic chemical reaction. To mimic the fiber-reinforced composites where fibers are aligned in a certain direction, thermal conductivity is assumed to be anisotropic. The thermal conductivity along the fiber direction k_{xx} and along the thickness direction k_{zz} is calculated as [58],

$$k_{xx} = v_r k_r + v_f k_f, \tag{8a}$$

$$\frac{k_{zz}}{k_r} = \left(1 - 2\sqrt{\frac{v_f}{\pi}}\right) + \frac{1}{B}\left[\pi - \frac{4}{\sqrt{1 - \left(B^2 v_f/\pi\right)}}\right] \tan^{-1} \frac{\sqrt{1 - \left(B^2 v_f/\pi\right)}}{1 + B\sqrt{v_f/\pi}}, \tag{8b}$$

$$B = 2\left(\frac{k_r}{k_f} - 1\right), \tag{8c}$$

where k_r and k_f are thermal conductivity values for matrix and fiber reinforcement, respectively; v_r and v_f are the resin and fiber volume fractions, respectively. The specific heat capacity and thermal conductivity are assumed to be constant during the curing process. A complete set of values of these parameters are given in the reference of [57,58]. Anandan et al. [58] implemented this thermochemical model in Comsol Multiphysics software to simulate the curing



(a) Schematics of the multi-physics solver (b) Solver validation with experimental data [58]

Fig. 2. Multi-physics finite-difference solver for the curing process: (a) schematics of the multi-physics thermochemical solver, and (b) validation of the solver with experimental data given the autoclave temperature profile as a boundary condition (BC). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

process. They modeled the tool and caul plate separately. Other consumable materials, such as vacuum bag, breather, and release film, were modeled as a single layer, assuming as a pure conductor with equivalent properties derived using the rule of mixtures.

2.3.2. *Numerical curing model for synthetic data generation*

The model equations described in Section 2.3.1 are solved numerically based on finite difference methods, and the numerical solver is implemented in PyTorch to generate synthetic data for training and validation. In the current study, only the laminate is modeled, and the manufacturing layups, such as tools, caul plate, and other consumable materials, are modeled as a single thin layer. Fig. 2(a) shows the schematics of this "ground truth" multi-physics finite-difference solver for the curing process, consisting of a coupled α-solver and T-solver solving the coupled dynamics of the cure kinetics and heat transfer based on forward Euler time stepping. Specifically, the cure kinetics equations (Eq. (4)) and energy equations (Eq. (6)) are solved iteratively to obtain the degree of cure α and temperature T inside the composite with the evolution of time. The spatial derivatives on the right-hand side of PDEs are discretized using the 2nd-order central differences. In order to validate the numerical solver, we conduct the simulation with the same setting provided in [58], where the experimental data is available. In particular, the laminate size of 304.8 mm $\times$ 25 mm is discretized using a mesh of 150 $\times$ 20 grid, and the numerical time-step Δt is set as 1 s. The validation results of the multi-physics finite-difference solver are shown in Fig. 2(b). Given the same autoclave temperature profile (black line), the simulated temperature (red line) is in good agreement with the experimental data (blue triangles), demonstrating the validity of the numerical solver.

In this study, we will use this validated multi-physics finite-difference model to simulate a virtual experimental environment and generate synthetic data for algorithm development, verification, and evaluation purposes. The composite laminate thickness ($\sim$ 25 mm) considered in [58] was small, and thus the spatial variation of temperature and degree of cure was insignificant, making it trivial for the data-driven modeling. To increase the complexity and make the problem more challenging, a thicker laminate (25 cm $\times$ 25 cm) is considered in this study. Specifically, the spatio-temporal data of the temperature and degree of cure for a composite laminate are generated by the numerical model, where the domain is discretized by a 30 $\times$ 30 mesh grid. The values of parameters used to generate synthetic data for training are given in Tables 1 and 2.

2.3.3. *PiNDiff model for curing process*

As mentioned above, composites manufacturing is a very complex process, and some of the physics might be missing. To imitate this situation, a showcase study will be conducted, where the analytical form of the cure kinetics (Eq. (4)) is assumed to be unknown, and some of the physical parameters, such as heat of reaction H_u, heat transfer

Table 1
Cure kinetic parameters for Cycom 5230–1 prepreg system [57,58].

Para	value	Para	value	Para	value	Para	value
A_1 (s^{-1})	1.48×10^7	A_3 (s^{-1})	6.39×10^7	A_2 (s^{-1})	8.3×10^4	A_4 (s^{-1})	9.8×10^4
$\frac{E_{A,1}}{R}$ (K)	1.02×10^4	$\frac{E_{A3}}{R}$ (K)	8.94×10^3	$\frac{E_{A2}}{R}$ (K)	8.54×10^3	$\frac{E_{A,4}}{R}$ (K)	7.1×10^3
m_1	0.17	m_3	1.65	m_2	0.7	m_4	1.66
n_1	19.3	n_3	16.6	n_2	0.87	n_4	3.9
D_2	97.4	D_4	63.3	$\alpha_{c0,2}$	-1.6	$\alpha_{c0,4}$	-0.6
				$\alpha_{cT,2}$ (K^{-1})	5.7×10^{-3}	$\alpha_{cT,4}$ (K^{-1})	3.0×10^{-3}

Table 2
Material parameters for Cycom IM7/5320–1 prepreg.

Para	value	Para	value	Para	value
ρ_c (kg/m^3)	1591.6	ρ_r (kg/m^3)	1310	ρ_f (kg/m^3)	1780
v_r	40.09%	v_f	59.91%	h (W/m^2K)	40
k_r (W/mK)	0.167	k_f (W/mK)	5.4	H_u (kJ/kg)	420
k_{xx} (W/mK)	3.3021	$k_{yy} = k_{zz}$ (W/mK)	0.5067	C_c	1260

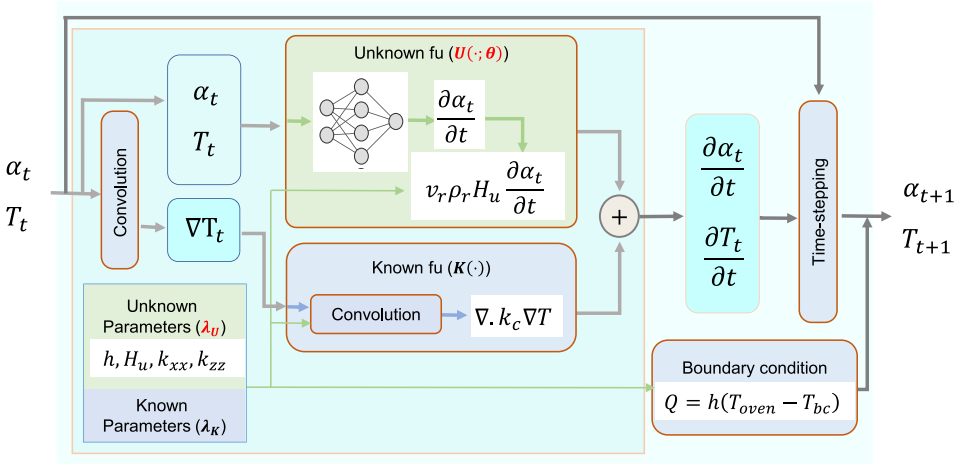


Fig. 3. Schematics of the PiNDiff module for the curing process.

coefficient h, thermal conductivity along fiber direction k_{xx} and along thickness direction k_{zz}, are considered missing as well. Therefore, only the energy conservation law is assumed to be known *a priori*, which will be integrated into the neural network to construct the PiNDiff model.[1] The schematics of the PiNDiff module designed for the curing process are shown in Fig. 3 where the known energy equation is encoded via non-trainable convolutions based on differentiable finite difference. A residual network component is designed to describe the cure kinetics, combined with the known PDE portions, to construct the hybrid neural solver. The pseudo-code for training the PiNDiff curing process model is given in Algorithm 1, which is similar to a generic neural network training procedure. Algorithm 2 shows the implementation of the PiNDiff neural solver for the curing process. Here, $NN_{kinetics}$ represents the neural network that approximates the unknown cure kinetics function $\mathcal{U}(\cdot)$. The detailed network architecture for the cure kinetics is given in Appendix (see Fig. C.23). The PiNDiff model can be trained with sparse and/or indirect data, e.g., only using the temperature labels at sparse spatial locations without any labels of the degree of cure. Moreover, the neural solver is also parametric. Once it is trained offline, the PiNDiff model can be used to make fast online surrogate predictions for new autoclave temperature boundary conditions, which are not seen during training, as discussed in Section 3. Normalization is critical in deep learning for better training efficiency and learning performance. For example, the outputs of the neural networks should be standardized to $\mathcal{O}(1)$ to facilitate

[1] Code will become available after publication at: https://github.com/Jianxun-Wang/PiNDiff-manufacturing.

convergence and increase the training speed. In this work, the residual network for the cure kinetics is normalized by the ratio of $|\alpha|/|t|$, where $|\cdot|$ represents either the data standard deviation if the data is available or the order of magnitude of the variable, which is assumed to be known *a priori*.

Algorithm 1 An algorithm for training PiNDiff curing process model.

Data: Read experimental label data
Initialize: Initialize neural network ($NN_{kinetics}$) and physical parameters (λ_U)
$epoch \leftarrow 0$
while $epoch < 300$ **do**
  $\alpha(x,t), T(x,t) \leftarrow Cure\_Solver(\alpha_{initial}, T_{initial}, T_{boundary}(t), \lambda_U, \theta)$
  $Pred \leftarrow Map\_to\_Observable(\alpha(x,t), T(x,t))$ ▷ Exp observables
  $L \leftarrow ||Pred - Label||_2$ ▷ Loss function
  $\theta \leftarrow \theta - \lambda \nabla L_\theta$ ▷ Update NN parameters
  if λ_U *need to be inferred* **then**
    $\lambda_U \leftarrow \lambda_U - \lambda \nabla L_{\lambda_U}$ ▷ Update λ_U parameters
  end
end

Algorithm 2 An algorithm for PiNDiff curing solver.

Function `Cure_Solver`$(\alpha_0, T_0, T_b(t), \lambda_U, \theta)$**:**
  $t \leftarrow 0$
  while $t < 400min$ **do**
    $\nabla \cdot k_c(\lambda_U)\nabla T_t \leftarrow Laplace\_Convolution(T_t; \lambda_U)$ ▷ Construct gradients
    $\dot{\alpha}_t \leftarrow NN_{kinetics}(\alpha_t, T_t; \theta)$ ▷ Predict cure kinetics using NN
    $\dot{T}_t \leftarrow \nabla \cdot k_c(\lambda_U)\nabla T_t + \nu_r \rho_r H_u(\lambda_U)\dot{\alpha}_t$ ▷ Integrated Known Physics $\mathcal{K}(\cdot)$
    $\alpha_{t+1}, T_{t+1} \leftarrow \alpha_t + \Delta t \dot{\alpha}_t, \quad T_t + \Delta t \dot{T}_t$ ▷ Time stepping, (can use RK4)
    $T_{t+1} \leftarrow Boundary\_Condition(T_{t+1}, T_b(t); \lambda_U)$
  end
  return $\alpha, T \leftarrow stack(\alpha_t, T_t)$ ▷ Predicted time series

3. Results

3.1. PiNDiff model: training with sparse and indirect data

Considering the fact that experimental data are often obtained at specific locations and may not be complete, we first evaluated the proposed PiNDiff model for scenarios where the training data is sparse and indirect. In [58], only the temperature at the center of the laminate was measured in their experiment. We followed the same setting and assessed the PiNDiff model, which was trained with the temperature time series at the center location based on the following loss function,

$$\mathcal{L} = \sum_{t=0}^{t_n} \|T_c - \hat{T}_c\|_2 + \beta_1 \|\theta\|_2 + \beta_2 \sum_{t=0}^{t_n-1} \|T(t+1) - T(t)\|_2 + \beta_2 \sum_{t=0}^{t_n-1} \|\alpha(t+1) - \alpha(t)\|_2$$

where T_c and $\hat{T}_c$ are predicted and measured temperatures at the center of the laminate.

The training was conducted on three different boundary conditions, i.e., autoclave temperature profiles. In particular, the autoclave temperatures of the three training cases are linearly increased with time, from 65 F (291 K) to three different values — case 1: 255 F (397.04 K), case 2: 265 F (402.59 K), case 3: 275 F (408.15 K), in 70 min, and then the oven temperature was held constant for 260 min (i.e., 70 min to 330 min) and finally dropped linearly back to 65 F (291 K) in 70 min. Namely, the training set consisted of three autoclave temperature profiles, varying linearly as [65, 255, 255, 65], [65, 265, 265, 65], and [65, 275, 275, 65] in F at the timestamps of [0, 70, 330, 400] in min, (as seen in Fig. A.19 in Appendix). Fig. 4 shows the PiNDiff prediction results given the autoclave temperature profile of case 2, where the model is trained for 900 epochs. The solid black line represents

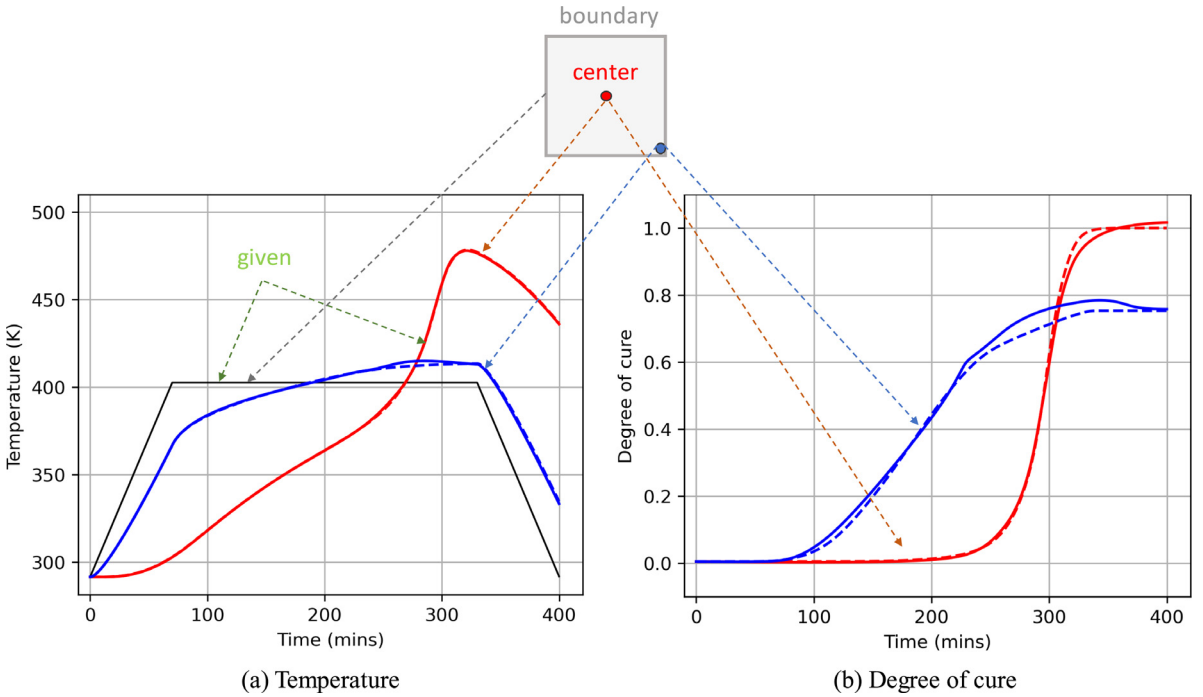


Fig. 4. PiNDiff predictions after training for 900 epochs on the temperature data collected at the center of the laminate. —— autoclave temperature (BC), —— prediction at center location, —— prediction at corner location, - - - ground truth at center location, - - - ground truth at corner location. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the time variation of the autoclave temperature, which is the input boundary condition to the model. The red and blue lines represent the temperatures at the center and corner locations of the laminate, respectively. It can be seen that the prediction of center temperature (solid red) perfectly matches with the ground truth (dashed red) since the corresponding labels were provided, demonstrating the great data fitting capability of the PiNDiff model. Although only the temperature history at the center of the laminate was provided as training labels, the trained PiNDiff model is also able to accurately infer the temperatures at other locations and even recover the whole field of the degree of cure, which is not observed experimentally (see Fig. 4(b)).

As a parametric neural solver, the trained PiNDiff model is also able to generalize to other different autoclave temperature inputs, which are not seen during training. Here, we tested the trained PiNDiff model on multiple out-of-training cases, where the autoclave profiles were randomly generated for testing purposes. Three of these test cases are shown in Fig. 5, where the autoclave temperature is varied linearly in piecewise as (Test case 1) [65, 140, 140, 265, 265, 265, 265, 148], (Test case 2) [65, 200, 200, 300, 300, 150, 150, 65], and (Test case 3) [65, 140, 140, 250, 250, 300, 300, 148] in F at the timestamps of [0, 30, 90, 130, 375, 400] in min. The prediction results are compared against the ground truth in Fig. 5.

It can be seen that the testing autoclave temperature profiles are significantly different from those used in training. However, both the predicted temperature and degree of cure are in reasonably good agreement with the ground truth. Even though the PiNDiff model is trained on very sparse data with a small batch size (i.e., only three different boundary conditions with a simple variation pattern), it shows excellent predictive capability and generalizability in the out-of-sample regimes. This example demonstrates the strong capability of the PiNDiff model in learning incomplete physics, inferring unobservable states, and predictive modeling of the curing systems with scarce, indirect measurements.

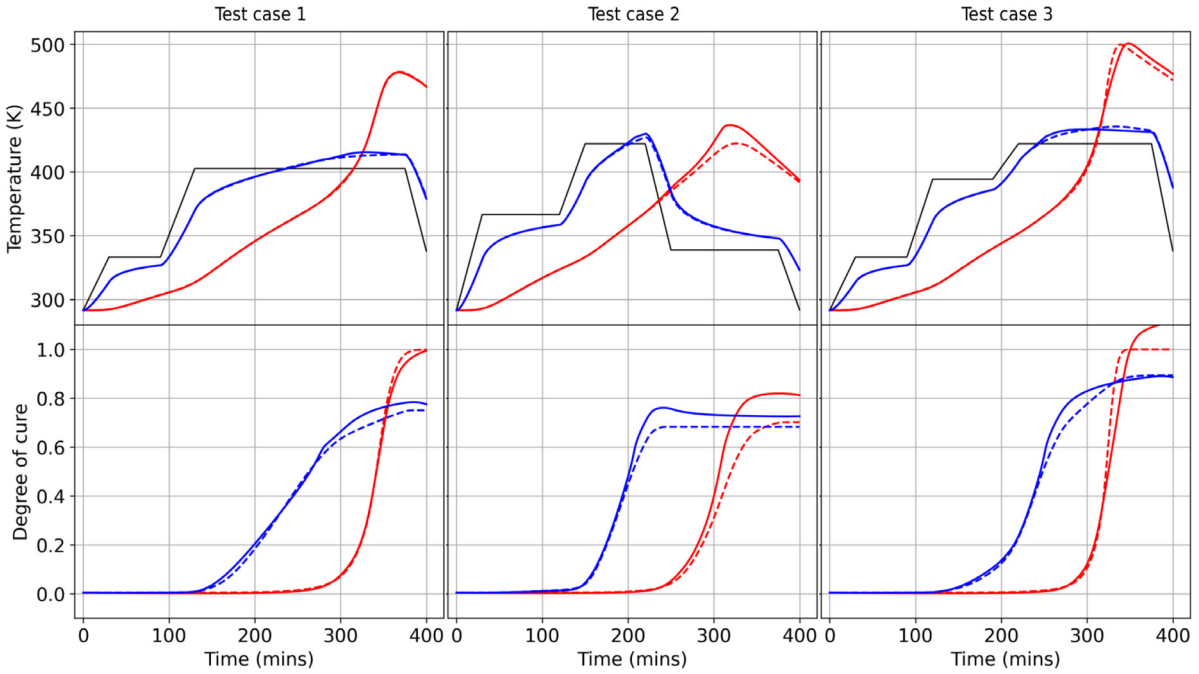


Fig. 5. PiNDiff model predictions for unseen testing autoclave temperature profiles, trained with temperature data only at the center. —— autoclave temperature, —— prediction at center location, —— prediction at corner location, - - - ground truth at center location, - - - ground truth at corner location.

3.2. Comparison with other SOTA deep learning models

To better evaluate the performance of the proposed PiNDiff model, we compare it with two existing SOTA purely black-box deep learning models based on either discrete recurrent neural networks or continuous neuralODE structures. Specifically, an Auto-Encoding Convolutional LSTM model (AE-ConvLSTM) [59–61] and NeuralPDE model [62] are also developed for the curing process modeling. The AE-ConvLSTM network contains two encoding ConvLSTM blocks and two decoding ConvLSTM blocks with 5×5 trainable kernels, and all the hidden connections are based on convolution operations. As for the NeuralPDE model, two separate neural networks are constructed: a fully-connected residual network for cure kinetics and a convolutional neural network for heat transfer modeling. The AE-ConvLSTM and NeuralPDE model have 510k and 447k trainable parameters, respectively, which are much higher than the 67k parameters of the PiNDiff model. The detailed architectures and hyperparameters of the two baseline networks are provided in Appendices B and C.

As shown in Section 3.1, the PiNDiff model can deal with incomplete and sparse data thanks to the integrated partially-known physics. That is, even though only a few temperature labels were used for training, the PiNDiff model could infer not only the full-field temperature but also the degree of cure, which is not observable. However, for most classic deep learning models, a large number of training labels are required for all the QoIs to be inferred during testing. The attempt to train these SOTA baseline models (i.e., AE-ConvLSTM and NeuralPDEs) with only the center temperature data will not work. Therefore, a complete dataset of temperature and degree of cure across the entire laminate plate (900 data points) were used to train the PiNDiff, AE-ConvLSTM, and NeuralPDE models in this comparison study. These training data were generated from the simulations with three different autoclave temperature profiles, the same as above, and the generalizability in testing scenarios will be compared.

3.2.1. Generalizability in input parameter space

We first compare the model's generalizability in the input parameter space once all three models are sufficiently trained with the entire dataset (900 training points). The models were tested on out-of-training input parameters, i.e., newly generated autoclave temperature profiles not seen in training. Fig. 6 shows the comparison results of

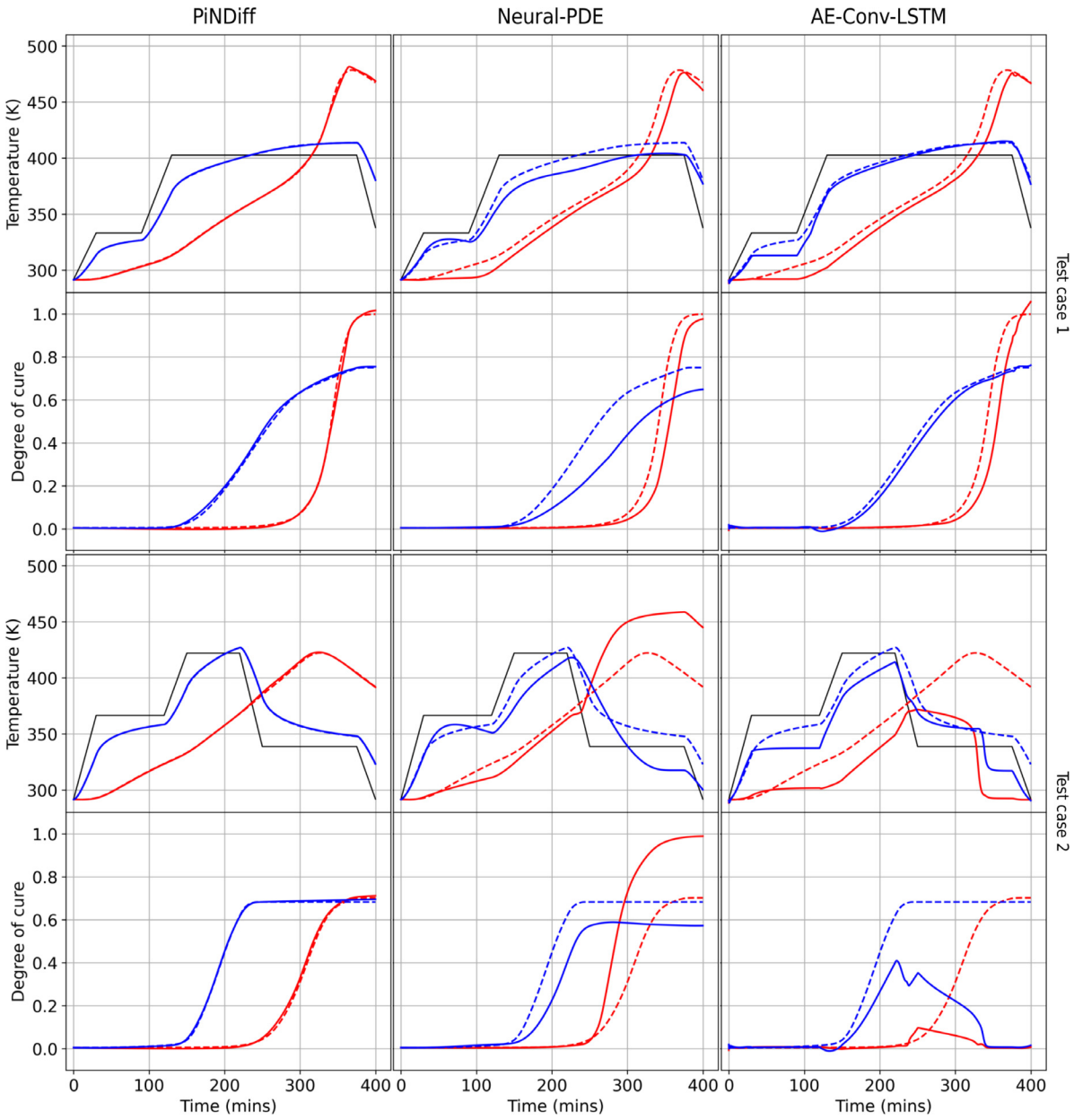


Fig. 6. Comparison of all three models on two unseen autoclave temperature profiles. —— autoclave temperature, —— predictions at center location, —— predictions at corner location, - - - ground truth at center location, - - - ground truth at corner location.

two representative test cases, where the autoclave temperatures are varied linearly in piecewise as (test case 1) [65, 140, 140, 265, 265, 265, 265, 148] and (test case 2) as [65, 200, 200, 300, 300, 150, 150, 65] in F with the timestamps of [0, 30, 90, 130, 375, 400] in min. The PiNDiff model accurately predicts the temperature and degree of cure for both cases, significantly outperforming the other two black-box SOTA deep learning models. For test case 1, where the BC temperature profile is close to those in the training scenario, AE-ConvLSTM prediction is slightly better than that of the NeuralPDE, whereas AE-ConvLSTM completely failed in test case 2, where the BC temperature profile is significantly different from training. This indicates that AE-ConvLSTM tends to overfit the training data. The NeuralPDE model performs better than AE-ConvLSTM as it can roughly capture the overall trends of the curing states, though with a much lower accuracy compared to the proposed PiNDiff model, owing

to its continuous residual connection formulation. In summary, the PiNDiff model has demonstrated a substantial generalizability advantage over the other two black-box neural networks, and thus it is well suited as a surrogate for parametric and optimization studies in out-of-sample regimes.

3.2.2. Generalizability with varying inference time intervals

To learn spatio-temporal physics, the model is usually trained on the snapshot data collected at a certain sampling frequency, i.e., fixed time interval Δt_{train} of the training snapshots. Most data-driven deep learning models, once trained on the time series data with a particular time interval Δt_{train}, can only make predictions with the same sampling frequency during the offline inference/testing, i.e., $\Delta t_{test} = \Delta t_{train}$, where Δt_{test} is the timestep used for testing. However, the training data could be temporally sparse in most real-world applications, and predictions with different time intervals from training (e.g., smaller timestep or higher sampling frequency) are often desired. Therefore, it is interesting to evaluate the model generalizability in terms of varying inference timestep sizes.

Figs. 7 and 8 show the comparison of model prediction results with sampling frequencies different from that of the training data. In particular, the timestep size Δt_{train} of the training time series is 10 s, while the trained model is evaluated with smaller time intervals, e.g., $\Delta t_{test} = 5$ s or 1 s. Fig. 7 shows the predictions with a training autoclave temperature profile, while Fig. 8 presents the results under an unseen testing autoclave temperature profile. It is clear that once the sampling frequency is changed, the AE-ConvLSTM model completely fails to capture the spatio-temporal fields, even for the training scenario. In contrast, both the NeuralPDE and PiNDiff models perform well when the testing time interval is reduced from 10 s to 5 s or even 1 s. This is due to the fact that the AE-ConvLSTM is a sequential net with a discrete recurrent network structure, and the time interval is fixed by construction, whereas both the NeuralPDE and PiNDiff models are based on the NeuralODE formulation, which is a continuous sequential residual network structure by solving the ODEs numerically, allowing time-integration with any timestep size. This advantage has been leveraged in dealing with corrupted and incomplete time series data [63]. Although both NeuralPDE and PiNDiff have better generalizability for inference time intervals, the proposed PiNDiff model has a higher prediction accuracy, especially in testing scenarios, as shown in Fig. 8, where a notable error of the NeuralPDE can still be seen, while the PiNDiff predicted curing states are almost identical to the ground truth due to the encoded physics.

3.2.3. Generalizability with spatial grids and geometries

Another aspect of the generalizability is how well the model can extrapolate to other shapes and mesh grids of the spatial domain. It would be ideal if the model trained on data from one geometry could be generalized to other spatial domains with different shapes. However, the two baseline SOTA networks involving convolutional operations are grid-dependent, and inference can only be made on the same mesh grids used in training since the input "image resolution" is fixed. Although these models can technically be evaluated on a new domain with a different aspect ratio if the grid topology remains the same, the predictions become nonphysical, as shown in Fig. 9, where all the models trained on a 25 cm $\times$ 25 cm laminate were tested on a 50 cm $\times$ 25 cm laminate. Both the NeuralPDE and AE-ConvLSTM failed due to the non-generalizable grid stretching of the spatial convolutions. In contrast, the proposed PiNDiff model can be well generalized to new domain shapes with high accuracy since the spatial convolution operations are based on known physics, which are not trainable, directly depending on the input geometries.

More testing results of the PiNDiff model on new laminate shapes with different aspect ratios and mesh grids are shown in Fig. 10. The training is conducted on a laminate with the shape of 25 cm $\times$ 25 cm, while the trained model is tested on the laminate shape of 10 cm $\times$ 50 cm, 5 cm $\times$ 75 cm, and 2.5 cm $\times$ 30.4 cm, with the grids of 10 $\times$ 60, 6 $\times$ 75, and 6 $\times$ 72, respectively. Moreover, the autoclave temperature profiles are not seen in training, making the model prediction more challenging. For all these cases, the PiNDiff predictions are in good agreement with the ground truth, showing excellent predictability and generalizability.

Figs. 11 and 12 show the contours of PiNDiff-predicted temperature and degree of cure fields on a laminate of size of 10 cm $\times$ 50 cm, compared with the ground truth. Both fields predicted by PiNDiff agree well with the ground truth. Although a high relative error is observed for the degree of cure predictions at initial steps, which is due to the small values in the first 150 min, the error drops steeply to be less than 1% after 200 min as the α value increases significantly. Therefore, the PiNDiff model can be trained with a laminate shape convenient for experimental settings, and then the trained model can be used for predictions of different shapes with reasonably good accuracy.

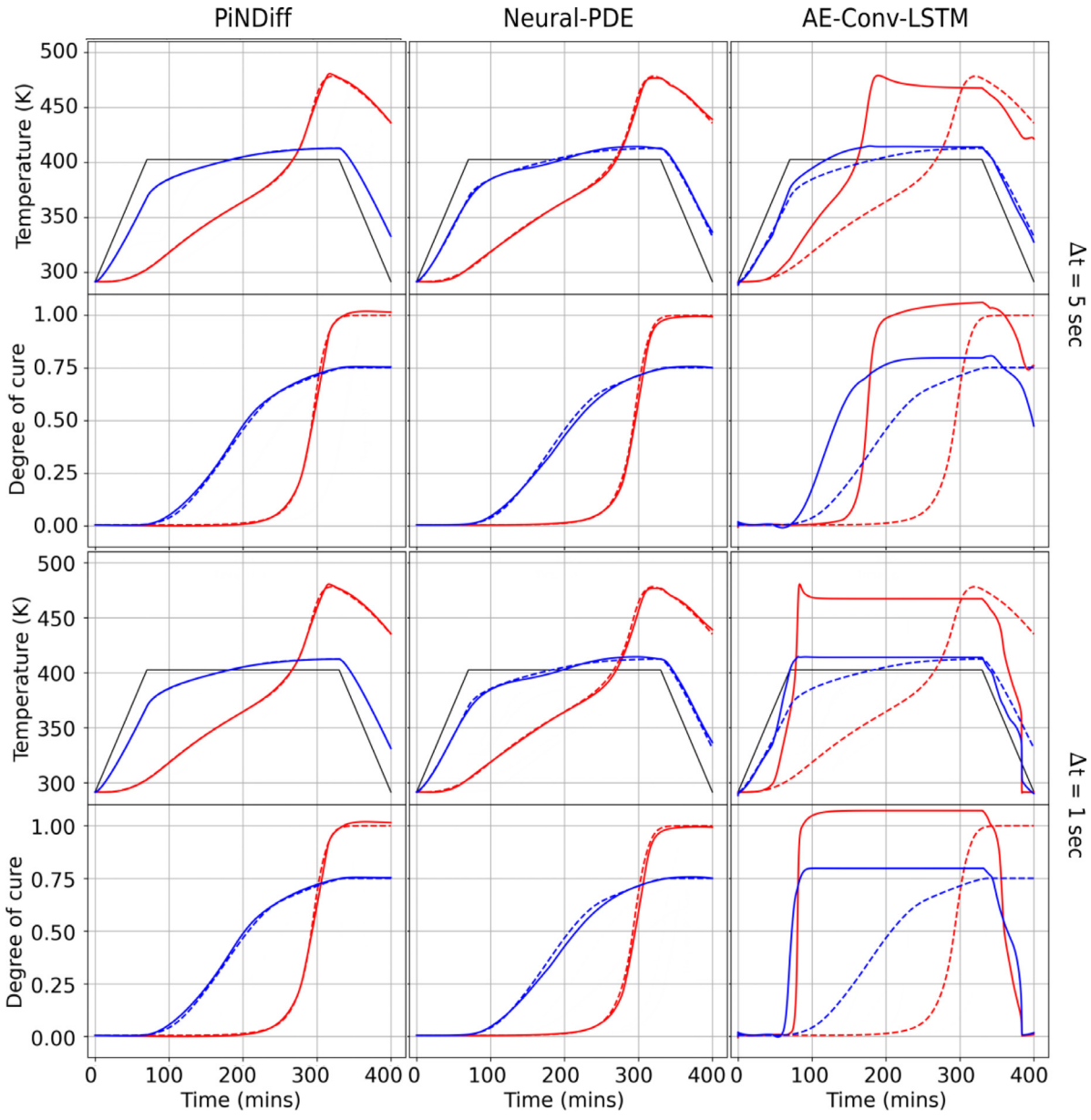


Fig. 7. Comparison of all three models on an autoclave temperature profile used for training, where the models are evaluated with varying timesteps Δt_{test} different from that used for training ($\Delta t_{train} = 10$ sec). —— autoclave temperature, —— predictions at center location, —— predictions at corner location, - - - ground truth at center location, - - - ground truth at corner location.

3.2.4. Computational cost

The prediction performance and computational costs of these models are listed in Table 3. Overall, the PiNDiff model has the lowest prediction error in both the degree of cure and temperature. The training cost for the PiNDiff model is nearly 3 h (epochs = 300), whereas the Neural-PDE and AE-Conv-LSTM require 8.5 h (epochs = 900) and over 20 h (epochs = 3000), respectively, to be sufficiently trained. As our PiNDiff (67K trainable parameters) is much lighter than NeuralPDE (447K trainable parameters) and AE-ConvLSTM (510K trainable parameters), the inference cost is only 18 s, which is lower than those of NeuralPDE and AE-ConvLSTM, which are 22 s and 30 s, respectively. Therefore, the proposed PiNDiff model is efficient in terms of training and inference, outperforming the two baseline SOTA deep learning models.

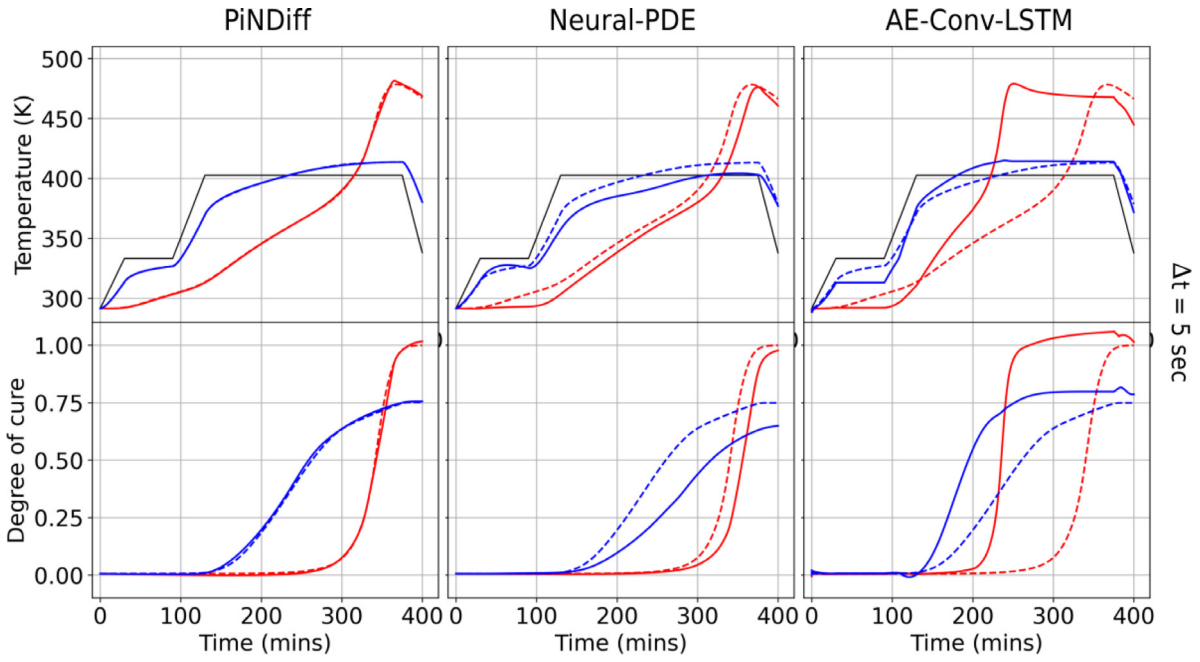


Fig. 8. Comparison of all three models on an unseen autoclave temperature profile, where the model are evaluated with a timestep of $\Delta t_{test} = 5$ s different from that in training ($\Delta t_{train} = 10$ s) —— autoclave temperature, —— predictions at center location, —— predictions at corner location, - - - ground truth at center location, - - - ground truth at corner location.

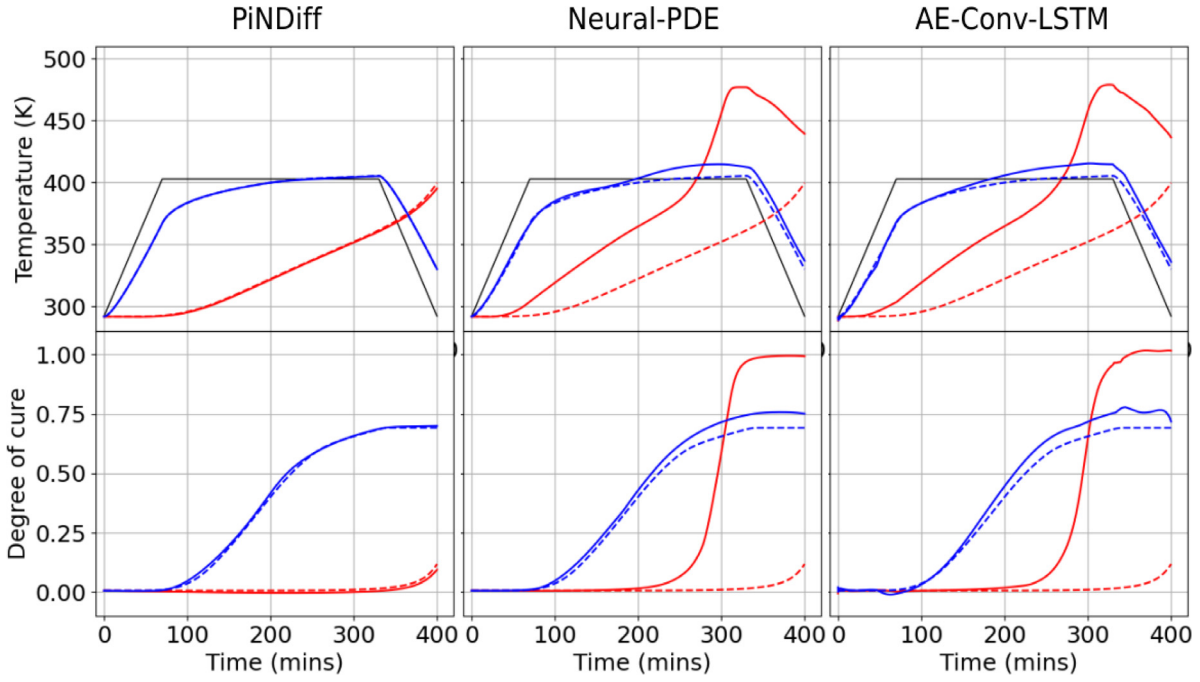


Fig. 9. Prediction results of a laminate size of 50 cm × 25 cm from PiNDiff, NeuralPDE, and AE-ConvLSTM models, trained on the data of a laminate size of 25 cm × 25 cm. —— autoclave temperature, —— predictions at center location, —— predictions at corner location, - - - ground truth at center location, - - - ground truth at corner location.

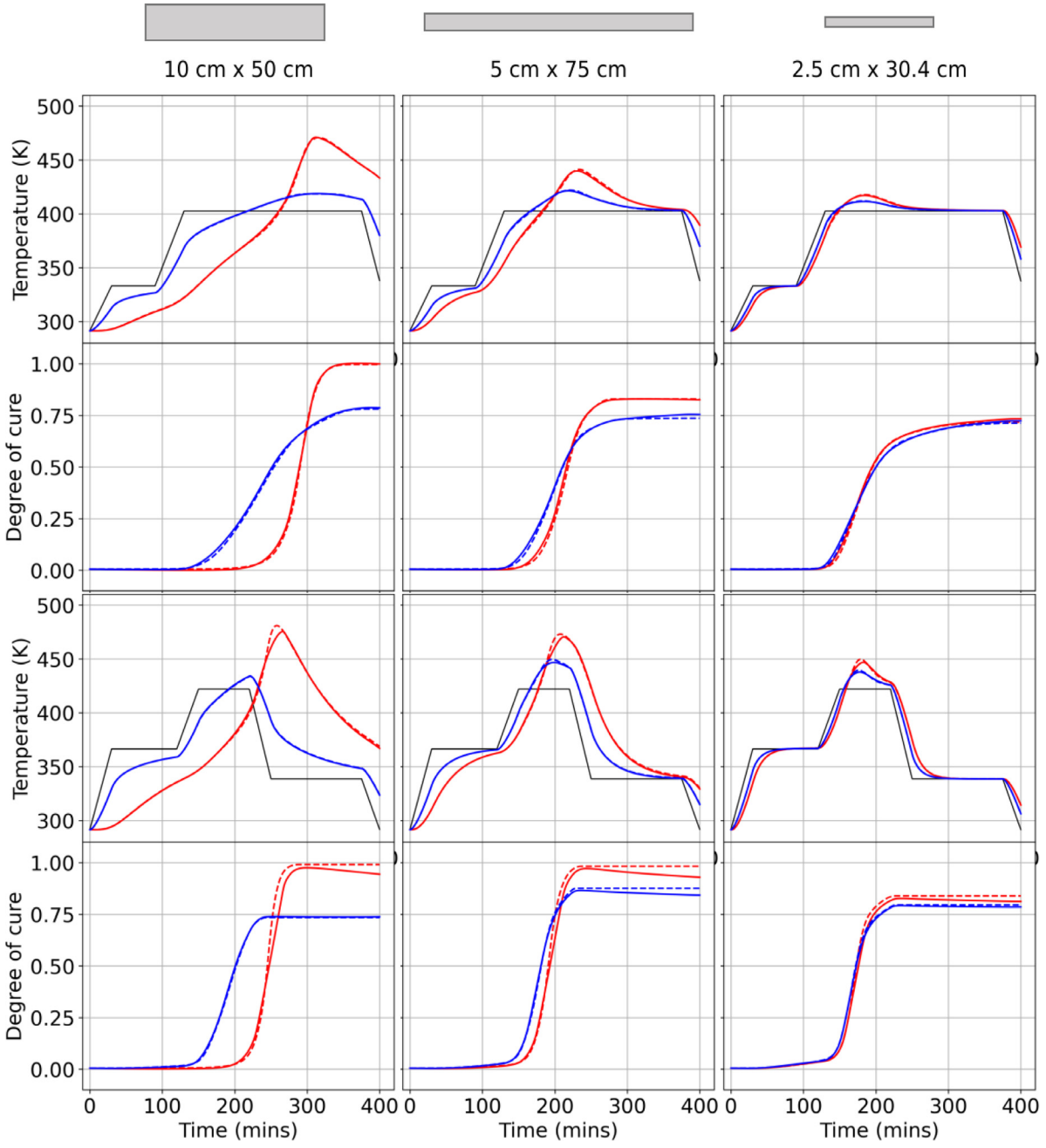


Fig. 10. PiNDiff model predictions on different laminate size (trained on laminate size of 25 cm × 25 cm) —— autoclave temperature, —— prediction at center location, —— prediction at corner location, - - - ground truth at center location, - - - ground truth at corner location.

PiNDiff as a fast surrogate model: in current problem formulation, the focus is to demonstrate the learning capability considering scenarios that the underlying physics is only partially known and thus to construct a classic finite-difference solver is not feasible. When the curing physics is completely known, the inference time of the PiNDiff model presented above is similar to that of the ground truth finite-difference solver, i.e., 18 s, which is due to the fact that the computational costs for computing cure kinetics using true analytical formulation (if the physics is fully known) and neural networks are similar. However, the PiNDiff framework can be formulated to result

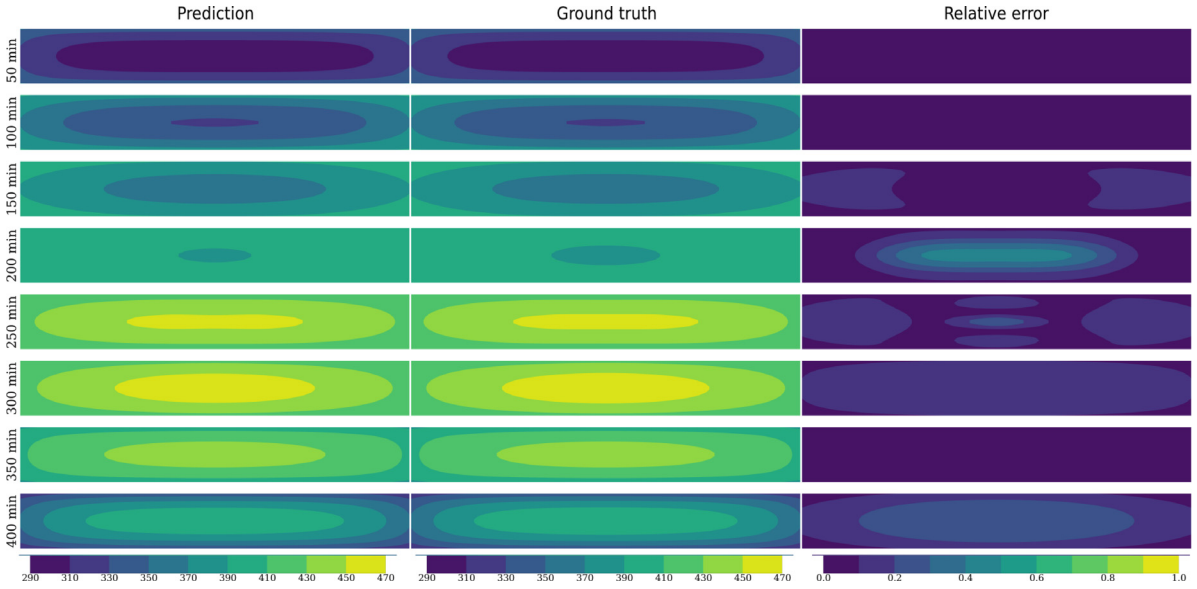


Fig. 11. Comparison of PiNDiff model temperature prediction and ground truth contours on laminate size of 50 cm × 10 cm (trained on laminate size of 25 cm × 25 cm).

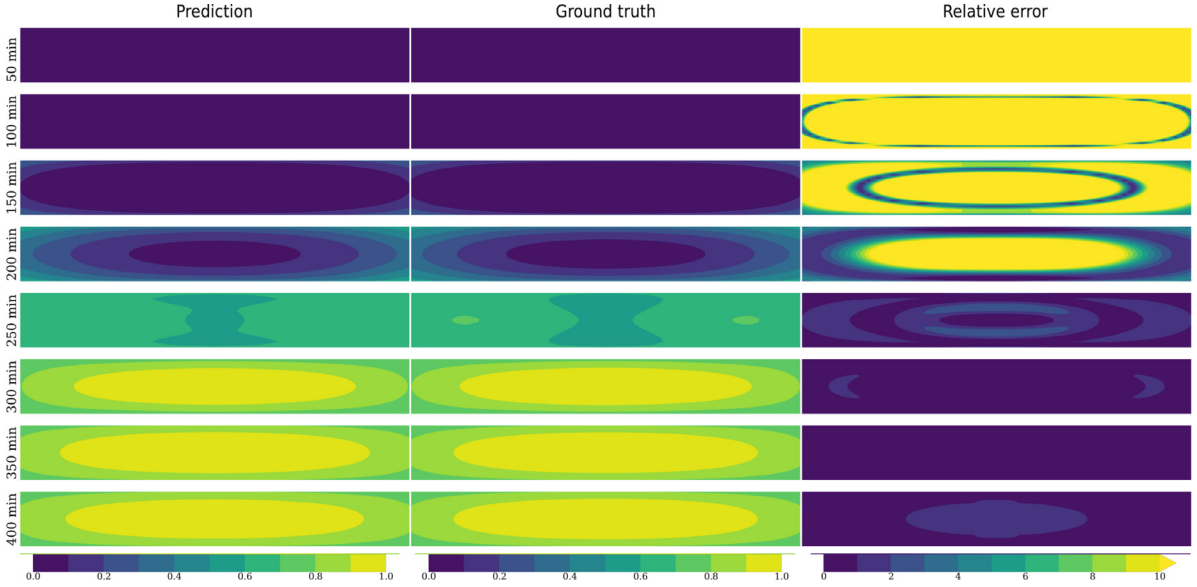
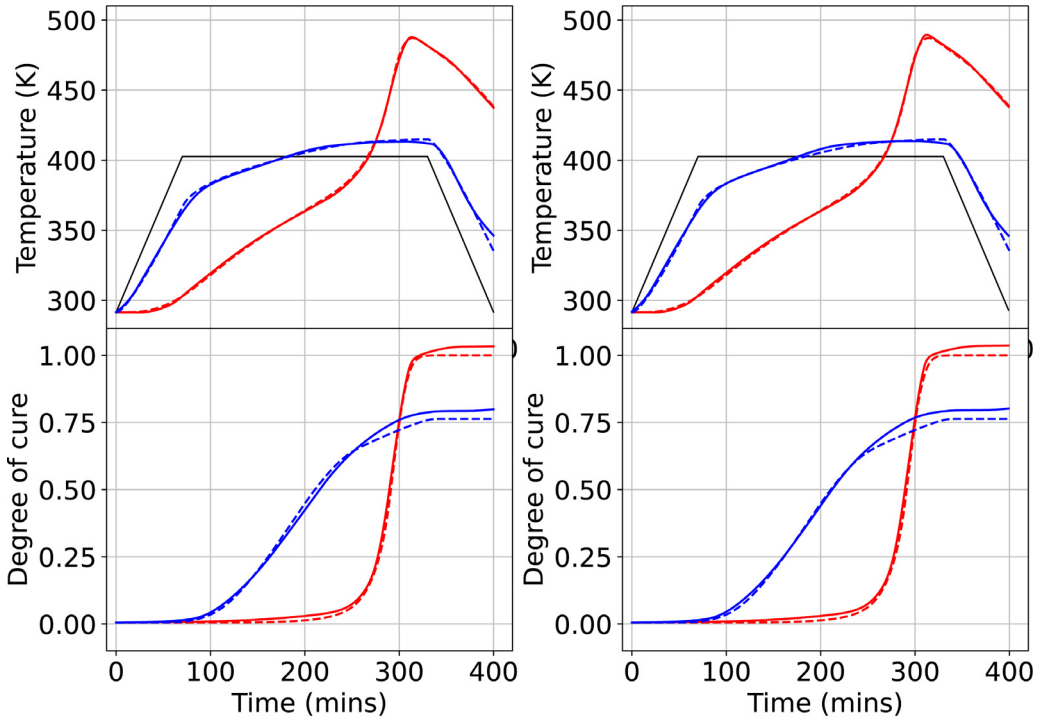


Fig. 12. Comparison of PiNDiff model degree of cure prediction and ground truth contours on laminate size of 50 cm × 10 cm (trained on laminate size of 25 cm × 25 cm).

in a fast surrogate model by replacing a time-consuming part (e.g., an elliptic solver) or releasing the numerical constraints (e.g., stability constraints) using deep neural networks. In order to demonstrate the speedup capability of the PiNDiff method, a different formulation is studied where both the heat diffusion term $\nabla \cdot k_c \nabla T$ (that constrains the solver) and cure kinetics are modeled by neural networks. In particular, the heat diffusion term is approximated by a convolutional residual network, and this modification allows the model to work on a larger timestep Δt. The PiNDiff model is trained with 900 label data points of the degree of cure and temperature with a timestep size Δt_{train} of 10 s The trained model is then tested using a timestep Δt_{test} of 60 s, and the prediction results are presented in Fig. 13, showing a good agreement with the ground truth. The model inference time is recorded as 7 s,

Table 3
Prediction performance and computational costs of different deep learning models (MSE represents mean square error).

	PiNDiff	Neural-PDE	AE-Conv-LSTM
Degree of cure MSE	0.09×10^{-3}	0.67×10^{-3}	1.19×10^{-3}
Temperature MSE	0.50	14.12	17.92
# of epochs	300	900	3000
# of parameters	67k	447k	510k
Training time	3 h	8.5 h	$>$ 20 h
Inference time	18 s	22 s	30 s



(a) Train ($\Delta t_{train} = 10sec$) (b) Test ($\Delta t_{test} = 60sec$)

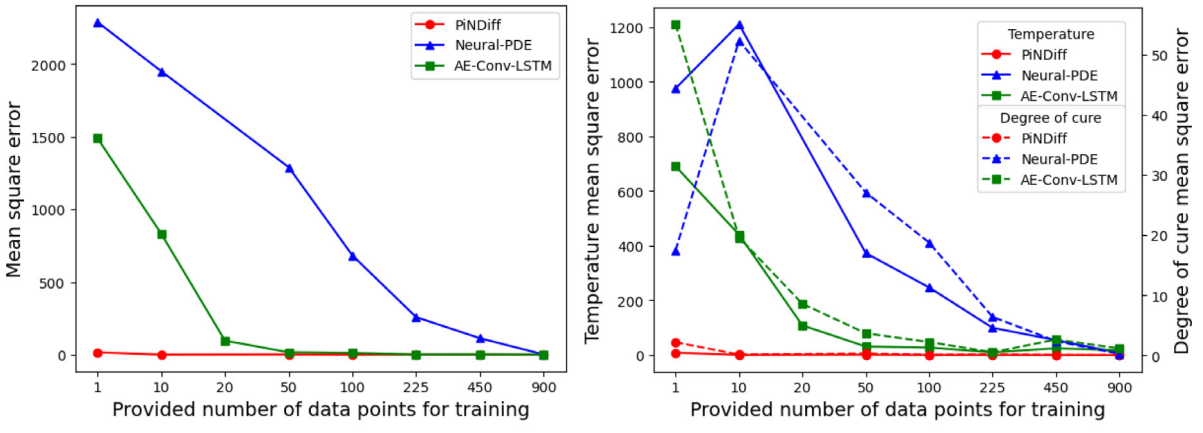
Fig. 13. PiNDiff model predictions during (a) training (Δt_{train} = 10 s) and (b) testing (Δt_{test} = 60 s) for fast surrogate study. —— autoclave temperature, —— prediction at center location, —— prediction at corner location, - - - ground truth at center location, - - - ground truth at corner location.

which is much less than that of the ground truth finite difference solver as 18 s It is noted that the classic solver cannot work with such a large timestep (Δt = 60 s). As the problem complexity increases, the speedup potential of the PiNDiff model becomes more significant.

3.2.5. Influence of training data size

Lastly, we would like to study the influence of the size of the training dataset. By varying the total amount of measured spatial points of the laminate, we compared the predictive performance of all three models under different data sparsity levels. In particular, the measured data points are randomly sampled across the entire laminate domain, from 900 points (complete set) to one point (minimal case), and the comparison results are plotted in Fig. 14 for (a) training and (b) testing scenarios.

The mean square errors (MSE) are computed based on testing cases with 50 randomly generated autoclave temperature profiles, which were not seen during training. These BC temperature profiles are generated by randomly sampling six temperature nodes ($T_i, i = 1 \cdots 6$) from a uniform distribution with the interval of [250 F, 300 F].



(a) Training (b) Testing

Fig. 14. Relative mean square error (MSE) of the model predictions versus the number of training data. The testing MSE is computed based on 50 randomly generated autoclave temperature profiles unseen in training.

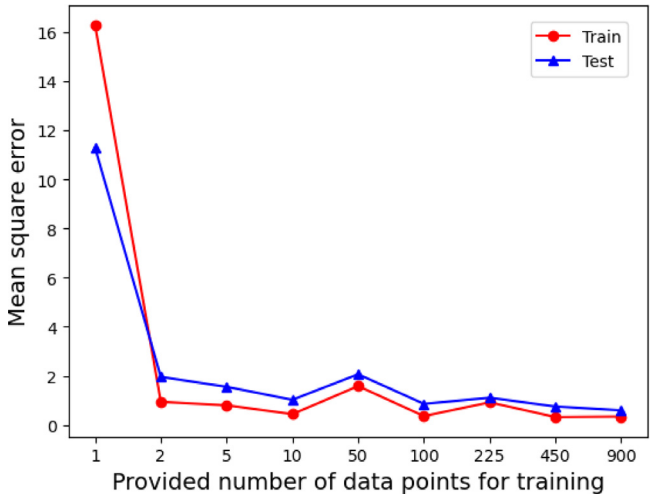
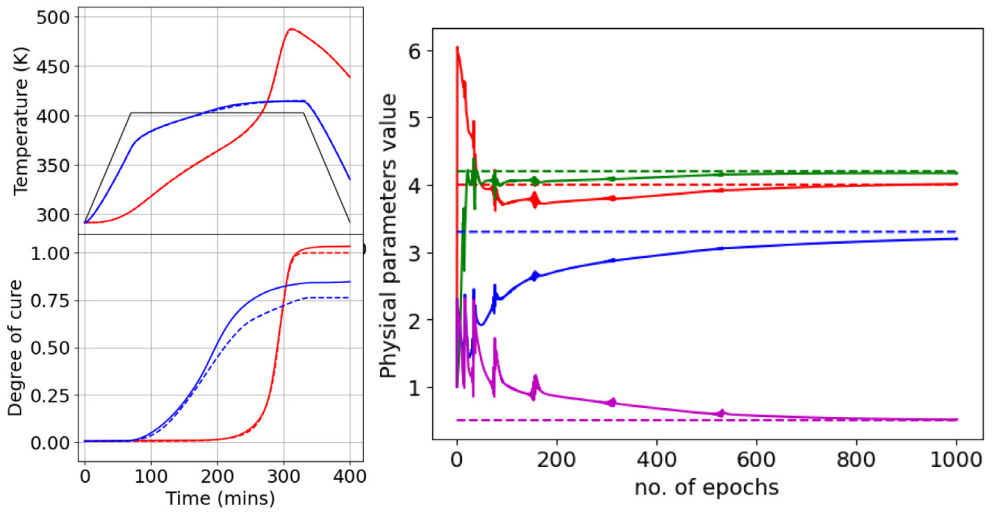


Fig. 15. Relative mean square error of the PiNDiff model versus the number of training data points.

The profiles are formed by linear interpolation between these nodes such that the temperature profiles vary as [65, T_1, T_2, T_3, T_4, T_5, T_6, 148] in F with timestamps of [0, 70, 122, 174, 226, 278, 330, 400] in min. The relative mean square error (MSE) for both training and testing cases decreases with the increase in training data points for both NeuralPDE and AE-ConvLSTM models. However, the MSE of the PiNDiff predictions remains low for all the sparsity levels even though only one data is used for training. As shown in the zoom-in view of the PiNDiff error curve (Fig. 15), only two training data points are sufficient for the PiNDIff model to reach a good predictive accuracy, attributed to the prior known physics assimilated into the network architecture. Whereas, the NeuralPDE and AE-ConvLSTM models will require much more data points to compete with the PiNDIff model. All the models work well when sufficient data is available for training, while the MSE of the PiNDiff model is still slightly lower than the other two (see Table 3). However, in sparse data regimes, it is clearly evident that the PiNDIff model significantly outperforms the other SOTA baselines.



(a) State predictions (b) Parameter inference

Fig. 16. PiNDiff prediction results with simultaneous physical parameter inference: (a) curing state prediction and (b) physical parameter convergence history. Solid lines represent model predictions, while dashed lines are ground truth. fig a: (—— autoclave temperature, —— prediction at center location, —— prediction at corner location, - - - ground truth at center location, - - - ground truth at corner location) fig b: (—— H_u, —— h, —— k_{xx}, —— k_{zz}, - - - $\tilde{H}_u$, - - - $\tilde{h}$, - - - $\tilde{k}_{xx}$, - - - $\tilde{k}_{zz}$).

4. Discussion

4.1. Simultaneous inference of unknown physical parameters

As mentioned in Section 2, even though the model form is known, some associated physical parameters $\boldsymbol{\lambda}_U$ might not be available or cannot be directly experimentally measured. The hybrid neural-physics structure of PiNDiff enables simultaneous inference of these unknown physical parameters during the training process. To investigate this capability, we conducted a numerical study in which four physical curing process parameters, i.e., the heat of reaction H_u, heat transfer coefficient h, and thermal conductivity along fiber direction k_{xx} and thickness direction k_{zz}, are assumed unknown and need to be inferred. The ground truth values of these physical parameters were set as,

$$\tilde{\boldsymbol{\lambda}}_U = [\tilde{H}_u, \tilde{h}, \tilde{k}_{xx}, \tilde{k}_{zz}]^T = [4.2 \times 10^5, 4.0 \times 10^1, 3.302 \times 10^0, 0.5067 \times 10^0]^T,$$

which were not disclosed to the model training. Synthetic training data were generated from the multi-physics simulation given these true parameters. The PiNDiff model was initialized as $\boldsymbol{\lambda}_U^0 = [1.0 \times 10^5, 1.0 \times 10^1, 1.0 \times 10^0, 1.0 \times 10^0]$ and trained using the following loss function,

$$\mathcal{L} = \sum_{t=0}^{t_n} \|\mathbf{u} - \hat{\mathbf{u}}\|_{L2} + \beta_1 \|\boldsymbol{\theta}\|_{L2} + \beta_2 \sum_{t=0}^{t_n-1} \|\mathbf{u}_{t+1} - \mathbf{u}_t\|_{L2} + \sum_{t=0}^{t_n} \|\max(\alpha, 1.1) - 1.1\|_{L2}, \tag{9}$$

where the last term constrains the degree of cure α to be less than 1 by penalizing the violation. As for simultaneously inferring unknown parameters, the training becomes more ill-posed, and these additional constraints prevent the model from diverging at an early stage. Fig. 16 shows the state-parameter prediction results of the PiNDiff model trained with curing data from only nine uniformly distributed locations. The model was not only able to capture the curing states, i.e., temperature and degree of cure (Fig. 16(a)) but also inferred all four parameters accurately (Fig. 16(b)). After 1000 training epochs, all four parameters converge to the ground truth values (dashed lines). This study indicates that the PiNDiff model can simultaneously learn the physical functions/operators and infer unknown physical parameters with limited labels.

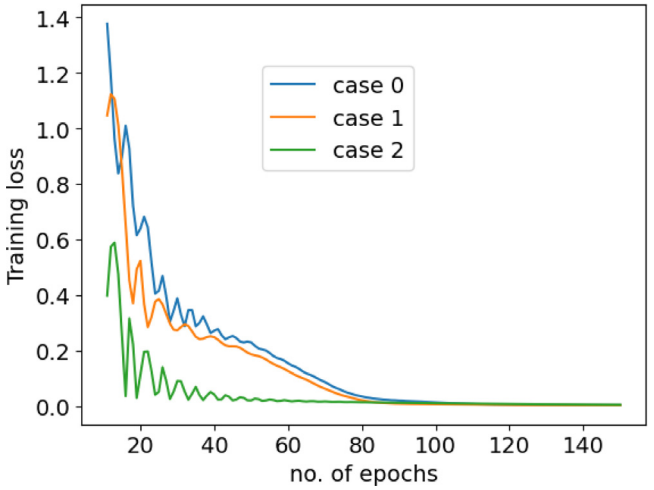


Fig. 17. PiNDiff training loss histories when (case 0) no cure kinetics physics are provided, (case 1) simple kinetics, i.e., first term on the RHS, is provided, and (case 2) more complex cure/temperature coupling, i.e., second term on the RHS, is provided.

Table 4
Mean square error of the three models with different levels of physics, trained with 900 labeled data.

Mean square error	Case 0	Case 1	Case 2
Degree of Cure	0.89×10^{-3}	0.73×10^{-3}	0.09×10^{-3}
Temperature	5.54	4.25	1.36

4.2. Importance of integrated prior physics

To illustrate the importance of prior physics integrated into the learning architecture, a study is conducted where different levels of known physics are provided. As discussed in Section 2.3, the cure kinetics described by Eq. (4) is treated as the complete "true" physics. As the cure kinetics equation involves complex chemical reaction mechanisms, which are usually unknown and was captured by neural networks, as shown above. Here, we are interested in situations where the cure kinetic physics is partially known. Therefore, three cases are considered where the cure kinetics equation is partially known.

case 0:	no RHS terms provided	$\frac{d\alpha}{dt} = \text{DNN}(\alpha, T)$
case 1:	1st RHS term is provided	$\frac{d\alpha}{dt} = \sum_{i=1,3} w_i K_i \alpha^{m_i}(1-\alpha)^{n_i} + \text{DNN}(\alpha, T)$
case 2:	2nd RHS term is provided	$\frac{d\alpha}{dt} = \text{DNN}(\alpha, T) + \sum_{j=2,4} w_j \frac{K_j \alpha^{m_j}(1-\alpha)^{n_j}}{1+\exp\left(D_j\left(\alpha-\left(\alpha_{C0,j}+\alpha_{CT,j}T\right)\right)\right)}$

The training loss histories for all three cases are plotted in Fig. 17. It is clear that without integrating any prior physics of cure kinetics, the training cost is the highest among the three cases. By leveraging partially known cure kinetics, the cost was reduced, and training was accelerated. As the first term on the right-hand side (RHS) of Eq. (4) is a simple function, whereas the second term is a more complex function of the degree of cure and temperature coupling, the training speedup of Case 2 is more significant than that of Case 1. Adding additional prior to the PiNDiff model further reduces the load on the neural network training as it only needs to learn less complicated functions.

Similarly, the prediction errors are computed by averaging the test cases with 50 randomly generated autoclave temperature profiles. Table 4 shows the comparison of the MSE for temperature and degree of cure predictions. It can be seen that lower prediction errors are expected as more physics is integrated, indicating increased generalizability.

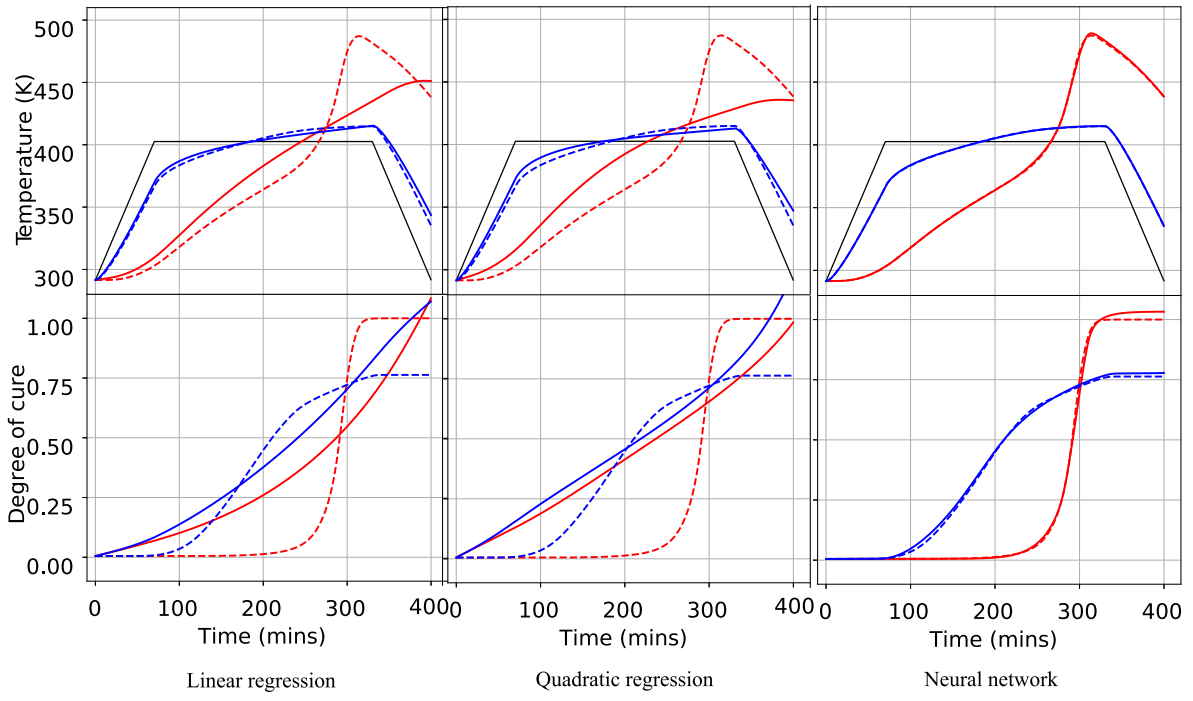


Fig. 18. PiNDiff model prediction with cure kinetics modeled as linear regression, quadratic regression, and neural network. —— autoclave temperature, —— prediction at center location, —— prediction at corner location, - - - ground truth at center location, - - - ground truth at corner location.

4.3. Neural network expressibility

The unknown $\mathcal{U}(\cdot)$ portions in the PiNDiff framework can be modeled as any machine learning regression model, such as linear or quadratic regression, as long as the entire program is fully differentiable. In this work, deep neural networks have been employed in the PiNDiff model to learn unknown $\mathcal{U}(\cdot)$ functions due to their strong expressibility and learning capability. To illustrate the importance of DNNs to learn unknown $\mathcal{U}(\cdot)$ functions, a study is conducted where the cure kinetics is approximated by other simple ML models (e.g., linear and quadratic regression) in place of DNNs within the same PiNDiff framework. In particular, the following regression models are used to learn the cure kinetics,

case 0:	linear regression	$\frac{d\alpha}{dt} = w_0 + w_1\alpha + w_2 T$
case 1:	quadratic regression	$\frac{d\alpha}{dt} = w_0 + w_1\alpha^2 + w_2 T^2$

The comparison of the prediction performances of all model variants with 900 training data for 1000 epochs is shown in Fig. 18. It is clear that these simple regression models (linear and quadratic regression) fail to learn the cure kinetics physics due to a lack of expressibility, whereas the DNN can well capture the complex physics even only with one labeled data as demonstrated in Section 3.1.

4.4. Limitations

Although the PiNDiff has shown great promise, there are still some limitations in its current form.

- Code intrusive: due to the nature of hybrid neural solvers, the PiNDiff model is highly code intrusive and the entire computer program should be fully differentiable, making the development and implementation challenging for complex physics described by nontrivial PDEs.
- Training stability: embedding physics in the PiNdiff network might result in training instability. In general, the physics module requires its inputs to be in its physically valid range to produce finite output, otherwise

resulting in the failure of the training process. The issue can be resolved by employing some constraints on the intermediate physical parameters inside the solver or limiting the temporal horizon for training.
- Geometry complexity: the current model uses a convolution operator for gradient calculation, limiting it to simple geometry and a structured grid. Replacing the convolution operator with a graph neural operator or using numerical methods for unstructured mesh gradient calculation will remove the limitation, and the model will work for complex geometry with unstructured grids.

5. Conclusion

In this work, a physics-integrated neural differentiable (PiNDiff) model was developed for simulating composite manufacturing processes where the underlying physics is partially known and available measurement data is limited and indirect. The key idea is to encode known physics in the form of discretized differential equations into DNN architectures within a differentiable programming framework. The hybrid neural solver has the capability to deal with data sparsity and better generalizability. The merit and effectiveness of the proposed PiNDiff model have been demonstrated in modeling the process of curing composite laminates, where partially-known physics, such as energy transport equations, are successfully integrated into a differentiable DNN architecture based on classic numerical techniques, e.g., finite difference and numerical time stepping, thereby requiring fewer trainable parameters and training labels. Compared with two existing SOTA black-box deep learning models, the PiNDiff model can work with "small data", and it shows significant superiority in terms of learning speed, generalizability, and robustness even when trained on massive amounts of data. Thanks to its fully differentiable structure, the PiNDiff model is able to infer the experimentally unobservable states and parameters during training. More broadly, the proposed PiNDiff framework can be used to effectively model a wide range of physical processes where the underlying physics is partially known, and sparse measurements are available.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Deepak Akhare, Tengfei Luo, Jian-Xun Wang reports financial support was provided by Air Force Office of Scientific Research. Jian-Xun Wang reports financial support was provided by National Science Foundation.

Data availability

A numerical model was used to generate data. The Git hub link will provide code.

Acknowledgments

The authors would like to acknowledge the funds from the Air Force Office of Scientific Research (AFOSR), United States of America under award number FA9550-22-1-0065. JXW would also like to acknowledge the funding support from National Science Foundation, United States of America under award numbers CMMI-1934300 and OAC-2047127, and startup funds from the College of Engineering at University of Notre Dame in supporting this study.

Appendix A. Detailed numerical setting for PiNDiff model

In PiNDiff model, the following learning setting is used for the neural network trainable parameters,

- Initial learning rate = 10^{-2}
- Optimizer = Adam
- Scheduler = ReduceLROnPlateau (factor = 0.7, patience = 20),

and for unknown physical parameter $\boldsymbol{\lambda}_U$ inference, the learning setting is listed as below

- Initial learning rate = 1
- Optimizer = Stochastic gradient descent (SGD)
- Scheduler = ReduceLROnPlateau (factor = 0.7, patience = 20).

The weighting parameters of the regularization terms β_1 and β_2 in $\mathcal{L}$ (Eq. (2)) are set as 10^{-8} and 10^{-5}, respectively.

A.1. Training autoclave temperature profiles:

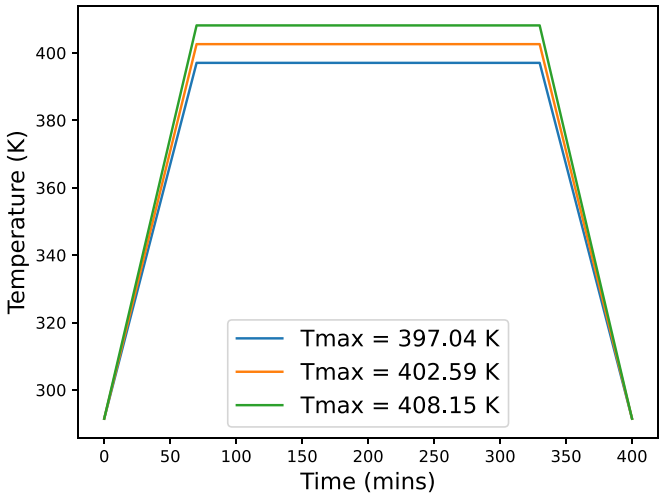


Fig. A.19. Autoclave temperature profiles (BC) of the training cases.

Appendix B. AE-ConvLSTM for curing modeling

The Encoder–Decoder ConvLSTM network architecture [60,61] is shown in Fig. B.20. The ConvLSTM network module is originally developed in [59] and the key equations used for calculating gate outputs are given as,

$$\begin{aligned} i_t &= \sigma\left(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i\right), \\ f_t &= \sigma\left(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f\right), \\ \mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh\left(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c\right), \\ o_t &= \sigma\left(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o\right), \\ \mathcal{H}_t &= o_t \circ \tanh\left(\mathcal{C}_t\right), \end{aligned} \tag{B.1}$$

where $*$ denotes the convolution operator, and $\circ$ denotes the Hadamard product. The convolution and de-convolution operations are applied to encode and decode spatial features, and ConvLSTM modules help to capture the temporal features, where all connections are based on convolution operations. By stacking these blocks in an auto-regressive manner with a ConvLSTM bridge, an AE-ConvLSTM is constructed. The hidden states produced by the ConvLSTM encoder are passed to the corresponding ConvLSTM decoder and next-step ConvLSTM encode, as shown in Fig. B.20 by gray arrows. The AE-Conv-LSTM network has the capability to learn spatio-temporal physics at various scales and make predictions of further states. The boundary condition can be enforced at the output layer of the network. The AE-ConvLSTM network predicts spatio-temporal fields in an auto-regressive sequence-to-sequence manner. In particular, the degree of cure and temperature fields are given as input and are predicted as output at each time step. For the current study, a two-block Encoder–Decoder ConvLSTM network is created with a 5×5 2D convolution kernel. The hidden channel size for the outer (i.e., first and last) ConvLSTM layers is 32, and for the inner (i.e., second and third) ConvLSTM layers, it is set to be 16.

Appendix C. Neural-PDE for curing modeling

Neural-PDE [62] network used in this paper is shown in Fig. C.21. The detailed neural network architecture used for heat transfer (T-ResCNN-net) and cure kinetics (α-ResANN-net) is shown in Figs. C.22 and C.23, respectively. The network architecture for cure kinetics is the same as the one used in the PiNDiff model. Note that α-ResANN-net is a Neural-ODE network since no spatial convolution is involved, as shown in Fig. C.23. The boundary condition is enforced at the output layer of the heat-transfer network. As shown in Fig. C.21, the α-ResANN-net takes the degree of cure and temperature as input to predict the degree of cure at the next time step and its derivative.

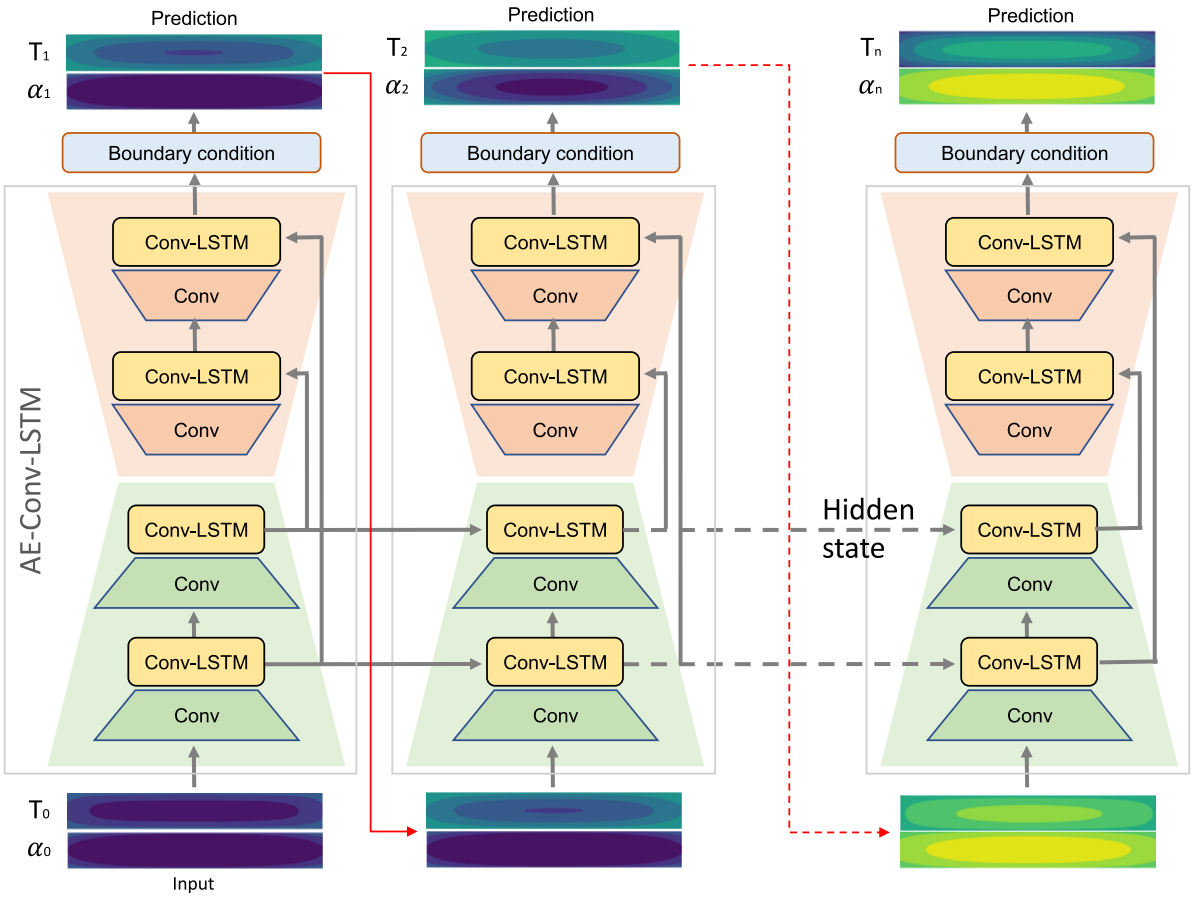


Fig. B.20. AE-Conv-LSTM network.

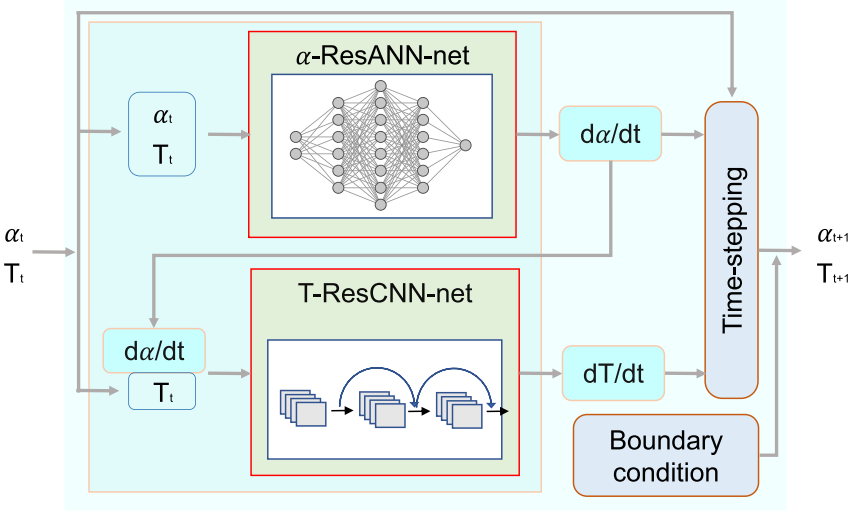


Fig. C.21. Neural-PDE network.

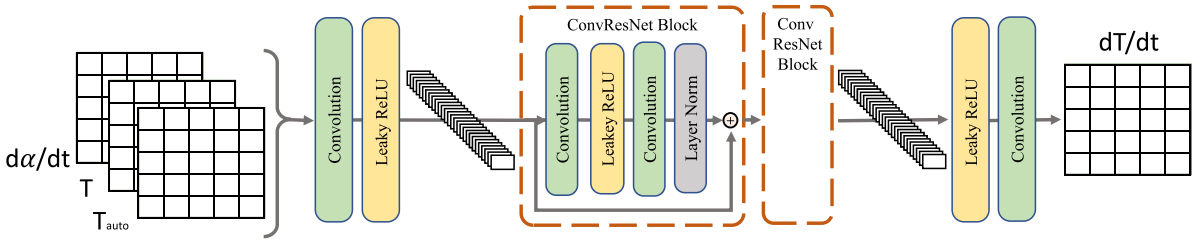


Fig. C.22. T-ResCNN-net for Neural-PDE model.

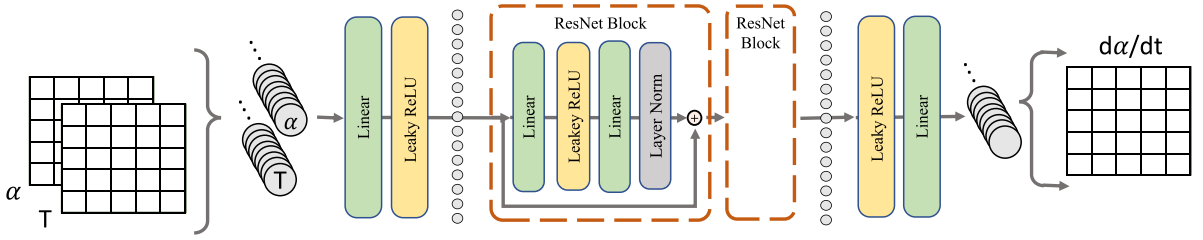


Fig. C.23. α-ResANN-net for PiNDiff and Neural-PDE model.

T-ResCNN-net takes the temperature and time derivative of the degree of cure to predict the temperature at the next time step. Therefore, the two sub-networks are fully coupled. Similar to PiNDiff and AE-ConvLSTM models, the NeuralPDE also predicts spatio-temporal fields auto-regressively. Each hidden layer of the α-ResANN-net (Linear in Fig. C.23) has 128 neurons and each hidden layer of T-ResCNN-net has 64 channels with a 3×3 trainable convolutional kernel.

References

[1] M.S. Sarfraz, H. Hong, S.S. Kim, Recent developments in the manufacturing technologies of composite components and their cost-effectiveness in the automotive industry: A review study, Compos. Struct. 266 (2021) 113864, http://dx.doi.org/10.1016/j.compstruct.2021.113864, URL https://www.sciencedirect.com/science/article/pii/S0263822321003251.

[2] Q. Fu, P. Zhang, L. Zhuang, L. Zhou, J. Zhang, J. Wang, X. Hou, R. Riedel, H. Li, Micro/nano multiscale reinforcing strategies toward extreme high-temperature applications: Take carbon/carbon composites and their coatings as the examples, J. Mater. Sci. Technol. 96 (2022) 31–68.

[3] I. Golecki, R. Morris, D. Narasimhan, N. Clements, Rapid densification of porous carbon–carbon composites by thermal-gradient chemical vapor infiltration, Appl. Phys. Lett. 66 (18) (1995) 2334–2336.

[4] Z.-h. Tang, D.-n. Qu, J. Xiong, Z.-q. Zou, Effects of infiltration conditions on the densification behavior of carbon/carbon composites prepared by a directional-flow thermal gradient CVI process, Carbon 41 (14) (2003) 2703–2710.

[5] J.-g. Zhao, K.-z. Li, H.-j. Li, C. Wang, The influence of thermal gradient on pyrocarbon deposition in carbon/carbon composites during the CVI process, Carbon 44 (4) (2006) 786–791.

[6] J. Kim, W.I. Lee, K. Lafdi, Numerical modeling of the carbonization process in the manufacture of carbon/carbon composites, Carbon 41 (13) (2003) 2625–2634.

[7] D. Mehta, A. Van Zuijlen, B. Koren, J. Holierhoek, H. Bijl, Large eddy simulation of wind farm aerodynamics: A review, J. Wind Eng. Ind. Aerodyn. 133 (2014) 1–17.

[8] M. Nakhchi, S.W. Naung, L. Dala, M. Rahmati, Direct numerical simulations of aerodynamic performance of wind turbine aerofoil by considering the blades active vibrations, Renew. Energy (2022).

[9] P.S. Ghatage, V.R. Kar, P.E. Sudhagar, On the numerical modelling and analysis of multi-directional functionally graded composite structures: A review, Compos. Struct. 236 (2020) 111837.

[10] K.H. Yang, J. Hu, N.A. White, A.I. King, C.C. Chou, P. Prasad, Development of numerical models for injury biomechanics research: a review of 50 years of publications in the stapp car crash conference, 2006.

[11] A. Hannam, Current computational modelling trends in craniomandibular biomechanics and their clinical implications, J. Oral Rehabil. 38 (3) (2011) 217–234.

[12] I. Martin, D. Saenz del Castillo, A. Fernandez, A. Güemes, Advanced thermoplastic composite manufacturing by in-situ consolidation: A review, J. Composites Sci. 4 (4) (2020) 149.

[13] P. Boisse, B. Zouari, A. Gasser, A mesoscopic approach for the simulation of woven fibre composite forming, Compos. Sci. Technol. 65 (3–4) (2005) 429–436.

[14] N. Hamila, P. Boisse, F. Sabourin, M. Brunet, A semi-discrete shell finite element for textile composite reinforcement forming simulation, Internat. J. Numer. Methods Engrg. 79 (12) (2009) 1443–1466.
[15] H. Tan, K.M. Pillai, Multiscale modeling of unsaturated flow in dual-scale fiber preforms of liquid composite molding I: Isothermal flows, Composites A 43 (1) (2012) 1–13.
[16] S.A. Niaki, A. Forghani, R. Vaziri, A. Poursartip, A three-phase integrated flow-stress model for processing of composites, Mech. Mater. 117 (2018) 152–164.
[17] S. Amini Niaki, A. Forghani, R. Vaziri, A. Poursartip, An orthotropic integrated flow-stress model for process simulation of composite materials—Part I: Two-phase systems, J. Manuf. Sci. Eng. 141 (3) (2019).
[18] S. Amini Niaki, A. Forghani, R. Vaziri, A. Poursartip, An orthotropic integrated flow-stress model for process simulation of composite materials—Part II: three-phase systems, J. Manuf. Sci. Eng. 141 (3) (2019).
[19] Y. Gao, J. Ye, Z. Yuan, Z. Ling, Y. Zhou, Z. Lin, J. Dong, H. Wang, H.-X. Peng, Optimization strategy for curing ultra-thick composite laminates based on multi-objective genetic algorithm, Composites Commun. 31 (2022) 101115.
[20] Z. Wang, W. Yang, Q. Liu, Y. Zhao, P. Liu, D. Wu, M. Banu, L. Chen, Data-driven modeling of process, structure and property in additive manufacturing: A review and future directions, J. Manuf. Process. 77 (2022) 13–31, http://dx.doi.org/10.1016/j.jmapro.2022.02.053, URL https://www.sciencedirect.com/science/article/pii/S1526612522001529.
[21] X. Liu, S. Tian, F. Tao, W. Yu, A review of artificial neural networks in the constitutive modeling of composite materials, Composites B 224 (2021) 109152, http://dx.doi.org/10.1016/j.compositesb.2021.109152, URL https://www.sciencedirect.com/science/article/pii/S1359836821005321.
[22] J. Huang, J. Liew, K. Liew, Data-driven machine learning approach for exploring and assessing mechanical properties of carbon nanotube-reinforced cement composites, Compos. Struct. 267 (2021) 113917, http://dx.doi.org/10.1016/j.compstruct.2021.113917, URL https://www.sciencedirect.com/science/article/pii/S0263822321003779.
[23] H. Chen, J. Yang, X. Chen, A convolution-based deep learning approach for estimating compressive strength of fiber reinforced concrete at elevated temperatures, Constr. Build. Mater. 313 (2021) 125437.
[24] M.H. Nguyen, R.J. D'Mello, A.M. Waas, Use of a neural network constitutive model for the size-dependent effects of curing on the deformation response and failure of fiber-reinforced polymer matrix composites, Arch. Appl. Mech. (2022) 1–19.
[25] I. Kopal, I. Labaj, J. Vršková, M. Harničárová, J. Valíček, D. Ondrušová, J. Krmela, Z. Palková, A generalized regression neural network model for predicting the curing characteristics of carbon black-filled rubber blends, Polymers 14 (4) (2022) 653.
[26] F. Tao, X. Liu, H. Du, S. Tian, W. Yu, Discover failure criteria of composites from experimental data by sparse regression, Composites B 239 (2022) 109947.
[27] K. Baek, T. Hwang, W. Lee, H. Chung, M. Cho, Deep learning aided evaluation for electromechanical properties of complexly structured polymer nanocomposites, Compos. Sci. Technol. (2022) 109661.
[28] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Engrg. 379 (2021) 113741.
[29] H. Gao, M.J. Zahr, J.-X. Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Engrg. 390 (2022) 114502.
[30] D.W. Abueidda, Q. Lu, S. Koric, Meshless physics-informed deep learning method for three-dimensional solid mechanics, Internat. J. Numer. Methods Engrg. 122 (23) (2021) 7182–7201.
[31] J.-X. Wang, J.-L. Wu, H. Xiao, Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data, Phys. Rev. Fluids 2 (3) (2017) 034603.
[32] X. Yang, S. Zafar, J.-X. Wang, H. Xiao, Predictive large-eddy-simulation wall modeling via physics-informed neural networks, Phys. Rev. Fluids 4 (3) (2019) 034602.
[33] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, Annu. Rev. Fluid Mech. 51 (2019) 357–377.
[34] E. Zhang, M. Dao, G.E. Karniadakis, S. Suresh, Analyses of internal structures and defects in materials using physics-informed neural networks, Sci. Adv. 8 (7) (2022) eabk0644.
[35] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks for heat transfer problems, J. Heat Transfer 143 (6) (2021).
[36] S.A. Niaki, E. Haghighat, T. Campbell, A. Poursartip, R. Vaziri, Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture, Comput. Methods Appl. Mech. Engrg. 384 (2021) 113959.
[37] R. Li, J.-X. Wang, E. Lee, T. Luo, Physics-informed deep learning for solving phonon Boltzmann transport equation with large temperature non-equilibrium, Npj Comput. Mater. 8 (2022) 19.
[38] N. Zobeiry, K.D. Humfeld, A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications, Eng. Appl. Artif. Intell. 101 (2021) 104232.
[39] M.P. Kaandorp, S. Barbieri, R. Klaassen, H.W. van Laarhoven, H. Crezee, P.T. While, A.J. Nederveen, O.J. Gurney-Champion, Improved unsupervised physics-informed deep learning for intravoxel incoherent motion modeling and evaluation in pancreatic cancer patients, Magn. Reson. Med. 86 (4) (2021) 2250–2265.
[40] A. Arzani, J.-X. Wang, M.S. Sacks, S.C. Shadden, Machine learning for cardiovascular biomechanics modeling: Challenges and beyond, Ann. Biomed. Eng. (2022) 1–13.
[41] M. Sarabian, H. Babaee, K. Laksari, Physics-informed neural networks for brain hemodynamic predictions using medical imaging, IEEE Trans. Med. Imaging (2022).
[42] S. Cuomo, V.S. 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, 2022, arXiv preprint arXiv:2201.05624.
[43] M. Raissi, P. Perdikaris, G. 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.

[44] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Comput. Methods Appl. Mech. Engrg. 361 (2020) 112732.
[45] R. Laubscher, Simulation of multi-species flow and heat transfer using physics-informed neural networks, Phys. Fluids 33 (8) (2021) 087101.
[46] A. Henkes, H. Wessels, R. Mahnken, Physics informed neural networks for continuum micromechanics, Comput. Methods Appl. Mech. Engrg. 393 (2022) 114790.
[47] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, Acta Mech. Sinica (2022) 1–12.
[48] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, J. Comput. Phys. 449 (2022) 110768.
[49] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V.B. Shah, W. Tebbutt, A differentiable programming system to bridge machine learning and scientific computing, 2019, arXiv preprint arXiv:1907.07587.
[50] B. List, L.-W. Chen, N. Thuerey, Learned turbulence modelling with differentiable fluid solvers, 2022, arXiv preprint arXiv:2202.06988.
[51] E. Heiden, D. Millard, E. Coumans, Y. Sheng, G.S. Sukhatme, NeuralSim: Augmenting differentiable simulators with neural networks, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 9474–9481.
[52] Y.-L. Qiao, J. Liang, V. Koltun, M.C. Lin, Scalable differentiable physics for learning and control, 2020, arXiv preprint arXiv:2007.02168.
[53] X.-y. Liu, H. Sun, J.-X. Wang, Predicting parametric spatiotemporal dynamics by multi-resolution PDE structure-preserved deep learning, 2022, arXiv.
[54] D.Z. Huang, K. Xu, C. Farhat, E. Darve, Learning constitutive relations from indirect observations using deep neural networks, J. Comput. Phys. 416 (2020) 109491.
[55] R.T. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, Adv. Neural Inf. Process. Syst. 31 (2018).
[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035, URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[57] D. Kim, T. Centea, S. Nutt, Out-time effects on cure kinetics and viscosity for an out-of-autoclave (OOA) prepreg: Modelling and monitoring, Compos. Sci. Technol. 100 (2014) 63–69, http://dx.doi.org/10.1016/j.compscitech.2014.05.027, URL https://www.sciencedirect.com/science/article/pii/S0266353814001742.
[58] S. Anandan, G. Dhaliwal, Z. Huo, K. Chandrashekhara, N. Apetre, N. Iyyer, Curing of thick thermoset composite laminates: multiphysics modeling and experiments, Appl. Compos. Mater. 25 (5) (2018) 1155–1168.
[59] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional LSTM network: A machine learning approach for precipitation nowcasting, Adv. Neural Inf. Process. Syst. 28 (2015).
[60] P. Ren, C. Rao, Y. Liu, J.-X. Wang, H. Sun, PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs, Comput. Methods Appl. Mech. Engrg. 389 (2022) 114399.
[61] P.R. Kakka, Sequence to sequence AE-convlstm network for modelling the dynamics of PDE systems, 2022, http://dx.doi.org/10.48550/ARXIV.2208.07315, URL https://arxiv.org/abs/2208.07315.
[62] Y. Sun, L. Zhang, H. Schaeffer, Neupde: Neural network based ordinary and partial differential equations for modeling time-dependent data, in: Mathematical and Scientific Machine Learning, PMLR, 2020, pp. 352–372.
[63] Y. Yang, M. Aziz Bhouri, P. Perdikaris, Bayesian differential programming for robust systems identification under uncertainty, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 476 (2243) (2020) 20200290.