An Advanced Physics-Informed Neural Operator for Comprehensive Design Optimization of Highly-Nonlinear Systems: An Aerospace Composites Processing Case Study

Anonymous Authors[1]

Abstract

Deep Operator Networks (DeepONet) and their physics-informed variants have shown significant promise in learning mappings between function spaces of partial differential equations (PDEs), enhancing the generalization of traditional neural networks. However, for highly nonlinear real-world applications like aerospace composites processing, existing models often fail to capture underlying solutions accurately and are typically limited to single input functions, constraining rapid process design development. This paper introduces an advanced physics-informed DeepONet tailored for such highly nonlinear systems with multiple input functions. Equipped with architectural enhancements like nonlinear decoders and effective training strategies such as curriculum learning and domain decomposition, the proposed model handles high-dimensional design spaces with significantly improved accuracy, outperforming the vanilla physics-informed DeepONet by two orders of magnitude. Its zero-shot prediction capability across a broad design space makes it a powerful tool for accelerating composites process design and optimization, with potential applications in other engineering fields characterized by strong nonlinearity.

1. Introduction

The simulation and optimization of engineering and scientific systems involves solving a set of partial differential equations (PDEs) across a range of system parameters. The study of these parametric PDEs involves obtaining the solution under different input functions such as initial condition (IC), boundary conditions (BCs), source functions, geometries, and coefficients. This entails repeated execution of computationally expensive numerical solvers such as finite element/volume methods (FEM/FVM) (Zauderer, 2011). Reduced-order methods have been introduced to enhance computational efficiency at the expense of reduced accuracy, offering a more practical solution for exploring the design space of parametric PDEs (Lucia et al., 2004). Similarly, recent advancements in deep learning, specifically in the field of scientific machine learning (SciML) empower the use of ML models for facilitating scientific discovery and optimization (Cuomo et al., 2022). SciML models infuse the unparallel approximation capabilities of ML models with the established governing physical laws to primarily accomplish one of the following tasks: 1) solve PDEs, 2) discover PDE parameters, and 3) learn operators. Methods such as physics-informed neural networks (PINNs) (Raissi et al., 2019) and deep Galerkin method (Sirignano & Spiliopoulos, 2018) have been introduced to approximate the solution of PDEs, however, they are inherently problem-specific and thus require retraining/finetuning in order to adapt to new system configurations. Although some enhancements have been introduced to improve their generalizability (Jagtap & Karniadakis, 2020; Gao et al., 2021; Ramezankhani & Milani, 2023; Majumdar et al., 2024), they still lack the adaptability required for the optimization tasks. Operator learning on the other hand targets the discovery of an unknown mathematical operator governing a PDE system (Boullé & Townsend, 2023). It seeks to capture a nonlinear mapping from one space of functions (inputs) to another space of functions (outputs). For a physical system described by PDEs, the input functions typically are the initial condition $u_0(x)$, boundary conditions $u_{bc}(t, x)$, and forcing term $f(t, x)$. The solution of PDE $u(t, x)$ is considered as the output function (Lu et al., 2022).

The operator is typically represented by a *neural operator*, a generalized form of neural networks, which can take functions (in a discretized form) as the inputs and outputs (Boullé & Townsend, 2023). Unlike PINNs, neural operators offer real-time prediction capabilities for varying system configurations without the need for retraining. This makes the

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

neural operator an ideal tool for performing parametric PDE studies. The training of neural operators, however, is computationally expensive and can incur poor generalization performance. Hence, several architectures have been introduced to tackle such shortcomings, including deep neural operator (DeepONet) (Lu et al., 2021), Fourier neural operator (FNO) (Li et al., 2020a), Graph neural operator (Li et al., 2020b) and Wavelet Neural Operator (WNO) (Tripura & Chakraborty, 2022), and deep Green network (Gin et al., 2021). These models are different in terms of discretization approach as well as approximation techniques employed for efficiency and scalability.

DeepONet, a neural operator method theoretically motivated by the universal operator approximation theorem (Chen & Chen, 1995), offers a more generalized operator learning framework to discover nonlinear function mappings. The two-part architecture of DeepONet consists of a branch net responsible for distilling the operator's input function into a fixed-size latent vector, and a trunk net, which decodes the output of the branch net to generate the final output at the specified locations. While DeepONet is primarily developed as a fully data-driven method, physical governing laws can also be incorporated to learn the solution operators in a fully physics-informed (i.e., data-agnostic) manner. Specifically, the existing governing equations and physical laws are integrated into the training of the DeepONet as additional loss terms (Wang et al., 2021). Despite the success of DeepONet and its physics-informed variant in effectively learning operators for a range of benchmark problems, it has been shown that the existing architectures may struggle to accurately capture the dynamics of PDE systems in complex real-life scenarios. In particular, for nonlinear submanifolds in function spaces, the finite-dimensional linear representation of DeepONet's decoder might be insufficient to learn the true target functions (Seidman et al., 2022). They also fall short when trained against multiphysics problems with coupled PDEs due to interactions between system variables, intricate geometries and lack of sufficient training data (Rahman et al., 2024). Furthermore, the physics-informed DeepONet (PIDON) also shares the same shortfalls as PINNs (Wang et al., 2022). Among them are failure to learn the long temporal domain, reduced accuracy against sharp edges and nonlinearities, and poor performance in multi-scale and multi-physics problems (Krishnapriyan et al., 2021).

This paper introduces an enhanced PIDON architecture capable of learning the solution operator mapping multiple input functions (i.e., design variables) to multiple output functions (i.e., decision variables) in a complex, highly nonlinear, and multi-physics engineering problem with a long temporal domain. Specifically, we explore the data-agnostic learning of the solution operator associated with coupled PDEs governing the thermochemical behavior of the aerospace-grade composites curing process in an autoclave under various input functions (i.e., process and material design configurations). Our investigation reveals the failure of DeepONets' original architecture to capture the complex dynamics inherent in the problem domain. To address this, we introduce a series of architectural improvements by integrating nonlinear decoders and employing multiple branch networks as well as leveraging advanced learning techniques such as curriculum learning and domain decomposition. We show that the proposed enhanced PIDON architecture can successfully learn the solution operator for this highly nonlinear PDE system and generate accurate predictions across various system configurations. While previous attempts have explored the application of data-driven neural operators (Chen et al., 2021; Rashid et al., 2022; Chen et al., 2023) and recently physics-informed FNO (Meng et al., 2023) in composites processing, their scope has been largely limited to singular design variables. Our work, on the other hand, broadens the horizon by incorporating multiple design parameters including but not limited to the cure cycle recipe, heat transfer coefficients (HTCs), and material thickness. This enhances the generalizability of neural operators, paving the path toward building scientific foundation models for more comprehensive and full-scale modeling and design optimization of complex engineering scenarios. Furthermore, we show that our enhanced PIDON outperforms previous models in terms of predictive performance.

The proposed framework refines the original DeepONet architecture to better tackle the inherent complexities present in real-world engineering scenarios, thereby enhancing its applicability and effectiveness. The contributions of this paper can be summarized as follows:

- Introduce an enhanced PIDON framework, featuring nonlinear decoders and multiple branch networks, to account for high nonlinearity and diverse input functions in complex PDE systems;
- Investigate the effectiveness of a series of remedies, such as curriculum learning and domain decomposition, typically used for efficient learning of PINNs, on the performance of PIDON;
- Develop a customized neural operator framework via introducing local spatial coordinates, enabling seamless learning of solution operators for the thermochemical curing process of composites in autoclave across multiple process design variables.

1.1. Application to advanced composites manufacturing

Superior mechanical properties, light weight and high durability have made fiber-reinforced polymer composites a popular choice of materials in high-performance applications where very large unified structures with intricate geometries are manufactured. Specifically in autoclave processing (Figure 6), a system of resin-impregnated fibers and a tool is

placed in an autoclave and subjected to a predefined temperature and pressure cycle (i.e., cure cycle) (Strong, 2008). The objective of this process is to cure the resin system in a way to achieve a uniform resin cure, optimum resin content, and a void-free product while minimizing any process-induced residual stress and deformation (Hubert et al., 2001). The through-thickness temperature profiles in the part and tooling as well as the evolution of the resin degree of cure are the key state variables of the composite system and crucial for foreseeing the process-induced defects. For any new scenario including modifications in part geometry and material properties, a process design optimization needs to be carried out via modifying a baseline cure cycle as well as optimizing the tooling and structure designs to ensure the above criteria are met. This iterative trial-and-error procedure is very expensive and time-consuming, especially for large structures, and thus prohibitive for real-world applications. Computational models have been developed as a more efficient alternative to model various aspects of the curing process such as heat transfer, resin cure kinetics, and residual stress development (Van Ee & Poursartip, 2009). While numerical methods such as FEM provide accurate approximations to the solution of PDEs, they can easily become computationally prohibitive when it comes to the iterative procedure of curing process optimization. The proposed PIDON model, on the other hand, can generate real-time predictions for various process configurations, significantly expediting the design optimization procedure. This enables a much faster exploration of the design space (e.g., processing scenarios, part and tooling designs) to identify the optimum process configurations for obtaining the desired material properties. In particular, following the terminology developed in (Fabris, 2018), for a composites manufacturing system with four main constitutive components (i.e., Process, Tools, Equipment and Parts), the following design variables can be handled by the proposed PIDON model (Figure 6):

- Process: cure cycle specifications including heating rate (r_1 and r_2), hold duration (hd_1 and hd_2), and hold temperature (ht_1 and ht_2)
- Tools and consumables: tool thickness (L_t)
- Equipment: convective HTCs in the autoclave (h_{top} and h_{bot})
- Parts: composite part thickness (L_c)

The parameters concerning Parts are typically predetermined in practice according to the application requirements. During the design optimization phase, the design engineer can conveniently select and fix the values of such variables and optimize the remaining design variables accordingly. Since the part and tool thicknesses are considered among the input functions (design variables) in the training of the neural operator and vary for each prediction task, it results in inconsistencies in the system dimensions as well as the composite-tool interface location (key for satisfying the continuity condition). In order to represent all training and test cases in a unified learning framework, we introduce two local spatial coordinates that independently describe the length of each material (Figure 1). The details regarding the governing equations of the thermochemical curing process as well as the implementation of local spatial coordinates are presented in A.1 and A.2.

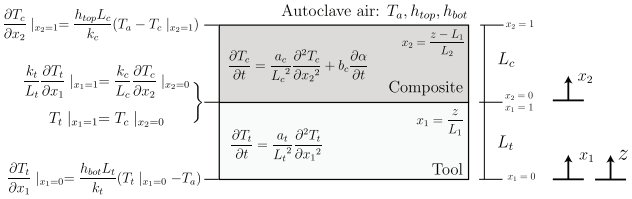

Figure 1. Schematic of composite-tool system in an autoclave with local coordinates x_1 and x_2.

The rest of the paper is structured as follows. Section 2 presents the proposed framework and discusses the constituents of the PIDON architecture. Section 3 is dedicated to an in-depth discussion of the results, investigating the performance of the proposed PIDON against the composites autoclave processing case study. Finally, the Conclusions section provides a summary and outlines future research directions.

2. Methodology

2.1. Physics-informed DeepONet

DeepONet's architecture is composed of a branch net and a trunk net. The branch net takes the sensor point evaluations $u = [u(x_1), u(x_2), \ldots, u(x_m)]$ as input and produces a finite-dimensional feature representation $b = [b_1, b_2, \ldots, b_q]^T \in \mathbb{R}^q$ as output. Similarly, the trunk net encodes the inputs of the PDE system y to a feature embedding $t = [t_1, t_2, \ldots, t_q]^T \in \mathbb{R}^q$ with the same size as the branch net's output. The output of the branch and trunk nets is then combined to calculate DeepONet's output using an element-wise product operation followed by a summation$G_\theta(u)(y) = \sum_{k=1}^{q} b_k t_k + b_0$. In a supervised learning fashion, the DeepONet can be trained by minimizing the error between the model's predicted output and the actual operator solution across a range of training input functions. Appendix B provides a detailed explanation of DeepONet's architecture along with a visual representation.

Drawing inspiration from PINNs, which learn the solutions of PDE systems by penalizing the residuals of the governing equations, a parallel approach is adopted in the development of PIDON framework (Wang et al., 2021). Specifically, the

output of DeepONet is constrained to align with the governing equations through the minimization of the loss function $\mathcal{L}(\theta) = \mathcal{L}_{IC}(\theta) + \mathcal{L}_{BC}(\theta) + \mathcal{L}_{physics}(\theta)$, where L_{IC} and L_{BC} are the IC and BC losses. Assuming a constant initial condition and a Robin boundary condition, L_{IC} and L_{BC} become

$$\mathcal{L}_{IC}(\theta) = \frac{1}{NQ_{ic}} \sum_{i=1}^{N} \sum_{j=1}^{Q_{ic}} |G_\theta(u^{(i)})(y_j^{(i)})) - s^{(i)}(y_j^{(i)})|^2 \tag{1}$$

$$\mathcal{L}_{BC}(\theta) = \frac{1}{NQ_{bc}} \sum_{i=1}^{N} \sum_{j=1}^{Q_{bc}} \left| \alpha G_\theta(u^{(i)})(y_j^{(i)}) + \beta \nabla G_\theta(u^{(i)})(y_j^{(i)}) - \gamma \right|^2 \tag{2}$$

where $u^{(i)}$ is the i-th input function, $y_j^{(i)}$ is the j-th collocation point in the operator domain and G_θ is the DeepONet output. $s^{(i)}(y_j^{(i)})$ is the initial loss represents the PDE solution at $y_j^{(i)}$ conditioned on the i-th input function. For the Robin boundary condition α, β, and γ are non-zero constants specified based on the physics of the problem. Similarly, $L_{Physics}$ is defined as:

$$\mathcal{L}_{physics}(\theta) = \frac{1}{NQm} \sum_{i=1}^{N} \sum_{j=1}^{Q} \left| \mathcal{N}(u^{(i)}(x), G_\theta(u^{(i)})(y_j^{(i)})) \right|^2 \tag{3}$$

where $\mathcal{N}$ is the nonlinear differential operator. In the above equations, N denotes the number of distinct input function combinations sampled from the design space and Q represents the number of residual points randomly sampled to enforce the physical constraints. They are considered hyperparameters and can be optimized based on the performance of the DeepONet and computational constraints.

2.2. Nonlinear decoder and multi-input functionality

The original architecture of DeepONet uses a linear decoder to learn the nonlinear operator. This results in approximating the target functions with a finite-dimensional linear subspace. However, for cases where the target functional data is concentrated in nonlinear manifolds, the vanilla DeepONet can easily fail unless a very high-dimensional linear decoder architecture is implemented (Lanthaler et al., 2022; Lee et al., 2023). This would result in a very large number of basis functions and coefficients determined by the output size of trunk and branch nets, respectively. Thus, the training of DeepONet can become computationally intensive for complex and non-smooth target functions. Different variants of DeepONet have been introduced to tackle the above limitation (Fang et al., 2024; Haghighat et al., 2024). One way to address this limitation is to integrate a nonlinear decoder (ND) into the architecture of DeepONet. Seidman et al. (2022) introduced NOMAD, a novel nonlinear manifold decoder leveraging neural networks to incorporate nonlinearity in the DeepONet's decoder. In this approach, the model merges the input function sensor values with the query points, forming a concatenated input, which is then fed to the decoder's network to predict the operator's output. Lee et al. (2023) proposed HyperDeepONet which substitutes the branch net with a hypernetwork to reduce the required network size for learning the solution operator. The hypernetwork is tasked with generating the weights of the trunk net given the input functions. Unlike the original architecture that distilled input function information into an embedding vector and fed it to the trunk net only in the final layer, the hypernetwork disperses this information at all layers of the trunk net. Here, while preserving the original architecture of DeepONet, we incorporate nonlinearity in the form of a fully-connected neural network immediately after merging the branch and trunk nets as depicted in Figure 2. Instead of summing the element-wise product of the trunk and branch networks' last layer, we channel the resulting vector to a neural network responsible for capturing the nonlinearity in the target functions space. We show that replacing the linear layer with a neural network as the ND not only improves the performance of DeepONet in complex PDE systems but also allows learning nonlinear operators with low-dimensional feature representations in the output of branch and trunk nets.

The DeepONet architecture is originally designed to map a single input function to the target output function. This imposes constraints, particularly in the process design optimization of engineering systems where there's a necessity to optimize multiple input functions (i.e., design variables) simultaneously with the optimization of engineering systems' process designs. To overcome this limitation, the proposed architecture utilizes a multi-input functionality as depicted in Figure 2, which allows the neural operator to effectively process multiple input functions. In particular, one branch network is dedicated to processing the time-dependent process parameters (BN2) while the second branch net is tasked to collectively process all time-invariant design variables (BN1). This is as opposed to assigning a separate network to each process parameter (Kumar et al., 2023), which significantly reduces the computational cost during the training of the DeepONet. The output layer size of the branch nets is the same as that of the trunk net. The branch nets' outputs b_1 and b_2 are merged via the Hadamard product, resulting in a q-dimensional vector $b = \sum_{i=1}^{q} b_i^1 b_i^2$. The resulting embedding which carries information about all input functions is then combined with the output of the trunk network,

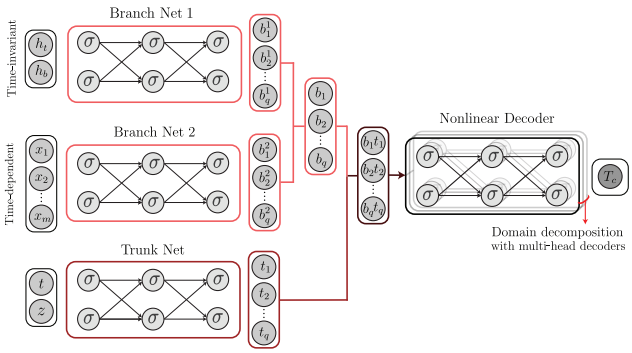

Figure 2. Schematic of proposed multi-input PIDON with NDs. Two branch nets are integrated to process time-dependent and time-independent input functions. NDs are responsible for learning the solution operators at different subdomains.

similar to vanilla DeepONet.

2.3. Domain decomposition

Similar to PINN, PIDON also enforces governing laws by incorporating physical constraints in the form of additional loss terms and minimizing them using optimization algorithms such as gradient descent or its variants. Thus, naturally, PIDON would inherit the limitations PINN encountered when learning highly nonlinear and non-smooth systems (Wang et al., 2022). The presence of nonlinear time-varying characteristics and sharp transitions (e.g., stiff PDEs) hugely deteriorates the performance of physics-informed models. Long temporal domains and the F-principle effect in neural networks are other common reasons for the failure of physics-informed models (Wang et al., 2024). Specifically, the curing process of composites in the autoclave occurs over a long period of time and involves highly nonlinear characteristics. To improve the performance of PIDON in such a system, we decompose the PDE domain into smaller subdomains and learn each subdomain using a separate ND. This is similar to sequential learning (Mattey & Ghosh, 2022; Wight & Zhao, 2020) and extended PINN approaches proposed for the training of PINNs (Jagtap & Karniadakis, 2020). As illustrated in Figure 2, we implement a multi-head decoder architecture where the NDs share the same branch and trunk nets, with each subnet dedicated to learning a segment of the time domain. The size of the subdomains depends on the physical characteristics of the problem at hand. By splitting the temporal domain into smaller intervals, we effectively break down the problem into multiple PDE instances, easier to handle by the NDs. The time intervals can be spread uniformly across the domain or selected according to the complexity of the problem, e.g., more intervals are concentrated around sharp transitions and less where the behavior is not as chaotic and nonlinear. Formally, we decompose the domain of the PDE problem Ω into N_d subdomains as $\Omega = \bigcup_{k=1}^{N_d} \Omega_k$. Each subdomain Ω_k is associated with an ND $f_k(\theta_k)$ tasked to learn the solution of the PDE within its subdomain. The continuity between the subdomains is enforced via an additional *interface* loss term, which is minimized along with the initial, boundary, and residual loss components during the training. The interface loss is calculated using the collocation points on the interface of adjacent subdomains $\partial\Omega_p \cap \partial\Omega_q$ where $i, j \in 1, 2, ..., N_d$. The full solution of the PDE problem in the domain Ω is achieved by combining all trained NDs. In this paper, the interface loss between the p-th and q-th subdomains is defined as:

$$\mathcal{L}_{IF}(\theta_p, \theta_q) = \frac{1}{NQ_{if}} \sum_{i=1}^{N} \sum_{j=1}^{Q_{if}} \Big| G_{\theta_p}(u^{(i)})(y_j^{(i)}) - G_{\theta_q}(u^{(i)})(y_j^{(i)}) \Big|^2 \tag{4}$$

2.4. Decoupled DeepONets for multi-output prediction

As elaborated in Section 1.1, the thermochemical analysis of composites during the curing process requires learning multiple target functions, namely, part temperature, tooling temperature, and part DoC. One way to achieve this is to devise multiple neurons in the output layer of DeepONet's ND, where each neuron is responsible for predicting one of the output functions. However, we observed that this configuration results in poor performance, mainly due to the significant discrepancies in the behavior of output functions and the limitation of the shared DeepONet architecture to capture those fully. Another approach would be to introduce a multi-head functionality where each output function has its own dedicated ND. The decoders take the branch and trunk networks' embedding as the shared input and separately learn the mappings to each target function. However, while more effective than the first strategy, it is yet unable to accurately learn the system's solution. We hypothesize that the presence of distinct thermal behaviors in this bi-material system as well as its multi-scale physics poses a challenge for a single branch-trunk network to effectively support the learning of the entire temperature and DoC fields. To address this, we utilize a fully decoupled DeepONet architecture where each output function has its own dedicated branch net, trunk net, and ND, as shown in Figure 3. This design yields three distinct neural operators denoted as G^{T_c}, G^{T_t}, and G^{α}. It is worth noting that decoupling the output functions automatically imposes a spatial domain decomposition as the temperature profiles of the part and tooling are learned via 2 separate DeepONets. This essentially allows some of NDs to concentrate exclusively on understanding the tooling's thermal characteristics, while the rest are dedicated to learning the thermochemical behavior of the composite

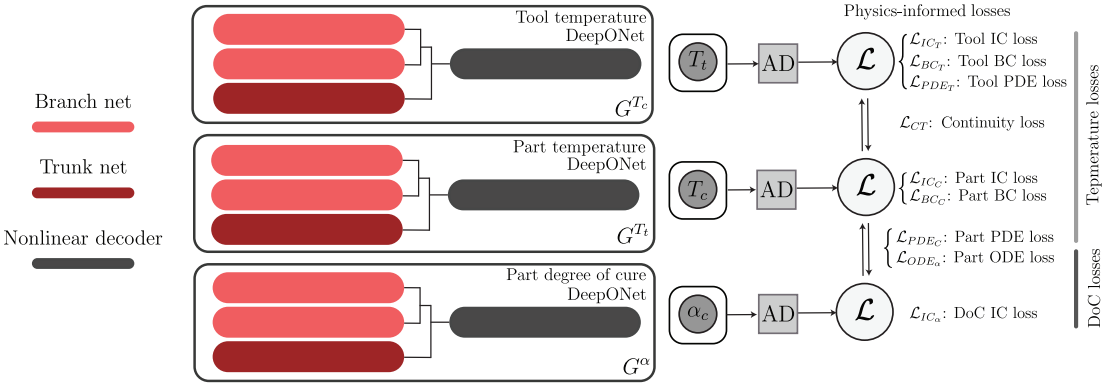

Figure 3. Schematic of proposed multi-input PIDON with nonlinear decoder for thermochemical analysis of composites curing process.

part.

2.5. Curriculum learning

It has been shown that the vanilla PINN model has difficulty learning highly nonlinear PDEs with large coefficients (Krishnapriyan et al., 2021). This can be due to the sensitivity of the PINN's loss landscape to the values of these coefficients. Particularly, when coefficients have large values, the loss landscape tends to become complex and asymmetric, posing significant challenges to the training process. To mitigate this issue, one effective approach involves utilizing curriculum learning strategies (Krishnapriyan et al., 2021; Bengio et al., 2009). By initially training the model on PDE solutions with smaller coefficients (i.e., smoother loss landscape) and gradually increasing the coefficient values, the model error can be reduced significantly by several orders of magnitude. We also observed similar challenges in training the PIDON model. Specifically, in the context of thermochemical analysis of composite curing processes, the presence of a heat generation term within the heat transfer governing equation introduces a sharp nonlinearity into the PDE solution (A.1). This nonlinearity substantially contributes to the poor predictive performance of physics-informed models. Particularly for large values of the heat generation coefficient (b_c), training the PIDON model becomes notably challenging. To address this issue, we employed the curriculum learning strategy described above. This involves initiating the training process with no heat generation term ($b_c = 0$) and gradually introducing internal heat generation to the equation through step-wise increments in the value of b_c.

3. Results

This section presents a series of experimental results demonstrating the effectiveness of the nonlinear decoders, curriculum learning and domain decomposition in the training of PIDON. The performance of PIDON against the highly nonlinear composites curing process across a high-dimensional design space is investigated. For model training and evaluation, 500 and 20 random combinations of input functions were generated from the specified ranges presented in Table 4. All branch, trunk and NDs of DeepONet models consist of 5 hidden layers with 50 neurons equipped with tanh activation function. A 50-neuron output layer is selected for both branch and trunk nets. PIDON was trained using Adam optimizer with an initial learning rate of 1×10^{-3} and a decay rate of 0.9 per 1000 steps. A batch size of 1024 and 200 training epochs was employed. The Jax library (Bradbury et al., 2018) was used for developing and training the models on a single NVIDIA T4 GPU with 104 GB of memory. For validation, an in-house Python FE code was developed and used to randomly generate unseen test cases from the design spaces. AS4/8552 prepreg and Invar tooling are considered as the materials for this case study. The model's average performance on these unseen test cases is reported. Details regarding the training procedure of PIDON models are presented in A.3.

3.1. Evaluation of PIDON's predictive performance

We trained the PIDON model on the design space characterized by the design variables outlined in Section 1.1. Three DeepONets equipped with NDs were trained to predict the output functions, specifically T_c, T_t, and α. The training of DeepONets was conducted sequentially (Niaki et al., 2021), where two Adam optimizers sequentially minimize temperature- and DoC-related losses (Figure 3) for improved stability and convergence. To capture the complex dynamics of the composite part, the time domain was decomposed into 7 intervals, with smaller intervals centered around the DoC sharp transition (Figure 7). The parametric coupled PDEs were learned via a curriculum learning strategy by incrementally increasing the heat generation coefficient b_c from 0 (no heat generation) to its real value in 5

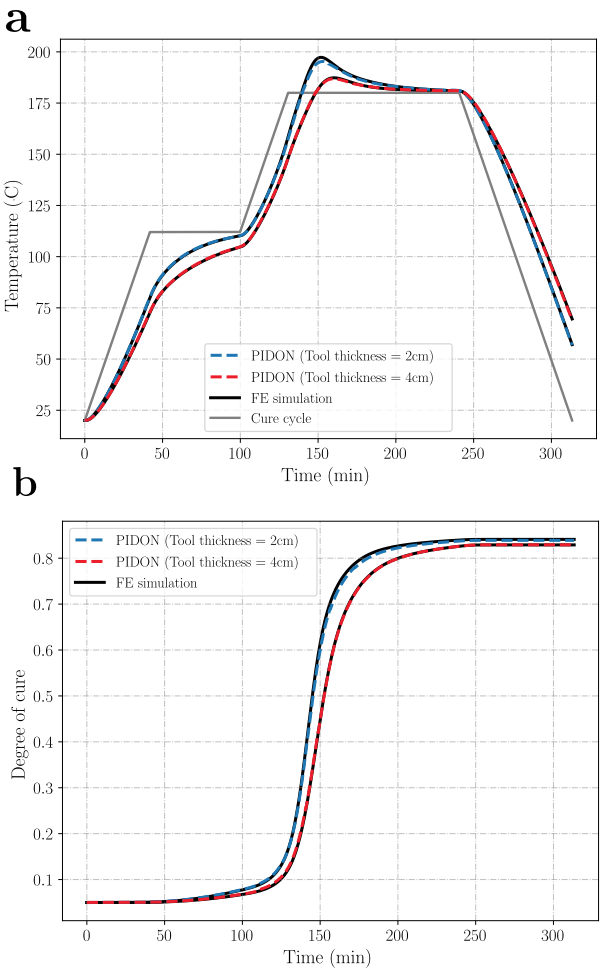

Figure 4. Temperature (a) and DoC (b) prediction performance of PIDON at composite part mid-point for two curing scenarios: thin tooling (T maximum absolute error = 3.2°C, α maximum error = 0.024) and thick tooling (T maximum error = 0.9°C, α maximum absolute error = 0.021). An identical two-hold cure cycle was used for both scenarios (shown in gray.)

steps.

Figures 4 and 9 illustrates the model's predictions of the part's mid-point temperature and DoC across various design variable combinations. The model achieved an average maximum absolute error of 2.3°C and 0.022 for temperature and DoC across test cases. Figure 10 provides a visual comparison between PIDON's predictions and FE simulations, along with the absolute error fields for part temperature, part DoC, and Tool temperature. The real-time inference capability of PIDON (20 times faster than FE simulations in this case) as well as its accurate predictions across a high-dimensional design space, makes it an excellent tool for process design optimization tasks.

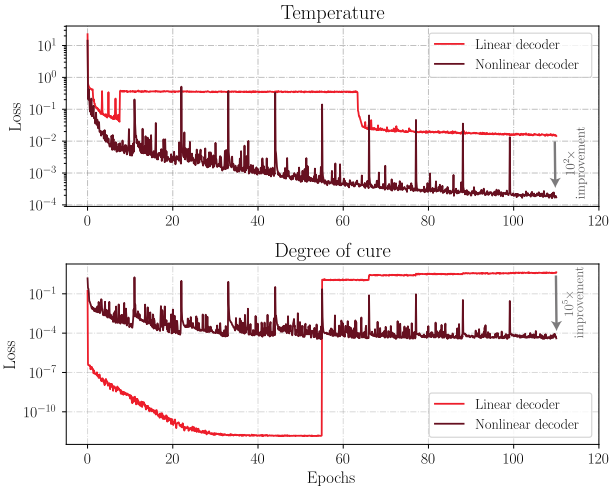

Figure 5. Effect of implementing ND in the architecture of DeepONet. The model with ND (dark red) results in considerably smaller training loss and exhibits a more stable behavior.

3.2. Effect of nonlinear decoder

To showcase the impact of integrating NDs into the architecture of PIDON, two training scenarios were considered: one with NDs and one without (using the original DeepONet's linear decoder.) As depicted in Figure 5, the model lacking the ND struggles to capture the complexities within the solution, ultimately converging to a relatively high training loss. Despite experimenting with various sizes of branch and trunk output layers within the vanilla DeepONet architecture, the model's performance remains unsatisfactory. In contrast, the PIDON model equipped with ND successfully learns the underlying physics, resulting in a significantly reduced training loss. Moreover, the addition of ND enables a more stable training process for learning the solution of the coupled PDEs, concurrently reducing both temperature and DoC losses to satisfactory levels. Conversely, PIDON's linear decoder initially converges to a trivial solution by equating the rate of change of DoC to zero (hence, a very small DoC loss at the initial stage of training). This, consequently, prevents the temperature loss from decreasing to small values. Upon exiting the trivial solution, while the temperature loss is improved, the DoC loss increases significantly and remains at large values. This observation underscores the inability of the linear decoder in the original architecture of DeepONet to learn nonlinear operators.

3.3. Curriculum learning

While utilizing NDs significantly improves the performance of PIDON, we still observed notable deviation between the PIDON's predictions and the FE solutions, particularly around sharp boundary edges and nonlinear trends during

Table 1. Comparison of PIDON's temperature prediction performance with and without curriculum learning across different design space sizes. The description of design spaces is provided in A.3. All models were trained using 7 NDs.

DESIGN SPACE SIZE	METRIC (AVG.)	REGULAR TRAINING	CURRICULUM LEARNING
SMALL	REL. L_2	6.8×10^{-3}	2.8×10^{-3}
	MAE	0.74	0.285
MEDIUM	REL. L_2	9.22×10^{-3}	3.27×10^{-3}
	MAE	0.83	0.361
LARGE	REL. L_2	1.3×10^{-2}	4.35×10^{-3}
	MAE	1.1	0.447

Table 2. Effect of Domain Decomposition on PIDON's temperature prediction performance in highly nonlinear regions.

METRIC (AVG.)	NUMBER OF NDS (N_d) 1	5	7
Rel. $L_2(\times 10^{-3})$	6.1	3.6	2.8
Max error (°C)	6.1	3.1	2.3
Training time (s/epc.)	40	56	61

the steep rise of DoC. This discrepancy is exacerbated as the problem complexity increases with broadening the input function range and expanding the design space. We hypothesize that one contributing factor is the nonsmooth, asymmetric, and complex loss landscape of the PIDON, hindering convergence to small loss values (Krishnapriyan et al., 2021). The presence of multiple loss components, including PDE, ODE, IC, BC, interface, and continuity losses compounds the optimization challenge. Here, we explored the impact of a curriculum learning strategy across three different design space sizes. Specifically, PIDONs were trained with and without curriculum learning on small, medium, and large input function ranges. For curriculum learning, training commenced with solving a simplified heat transfer problem without heat generation by setting the heat generation coefficient b_c (A.1) to zero. Then, the training gradually progressed to more complex scenarios by incrementally increasing the value of b_c in a step-wise fashion (Figure 11). At each stage, the trained weights from the previous step served as initialization. Table 1 presents a comparative analysis of PIDON performance with and without curriculum learning across various design space sizes. While PIDON accuracy diminishes with larger design spaces, those trained with curriculum learning consistently exhibit strong performance across all design space scales.

3.4. Role of domain decomposition in capturing nonlinearities

We conducted experiments to evaluate the efficacy of domain decomposition in enhancing the performance of PIDON for modeling the thermochemical analysis of composites curing process. Notably, while curriculum learning improved the model's overal performance, we observed that it still struggled to fully capture the behavior around sharp nonlinearities, particularly during the rapid increase in the DoC, responsible for the heat generation phenomena. This directly affects the model's performance on the prediction of exotherm (maximum part temperautre), a key factor in determining the quality of the manufactured part. To address this limitation, we utilized domain decomposition by partitioning the time domain into smaller intervals centered on the nonlinear region and allocating NDs to learn the physics within each interval. This approach further improved the model's performance around the nonlinearities. Specifically, we tested three architectures with different number of subnets as shown in Table 2. All models followed an identical training procedure, including the implementation of the curriculum learning strategy and an equal number of training epochs. The exotherm prediction error decreased substantially as the number of NDs increased. These results suggest that despite the effectiveness of curriculum learning in mitigating the adverse impacts of nonsmooth landscapes associated with large PDE coefficients, achieving more accurate predictions in highly nonlinear regimes necessitates more expressivity, which can be attained through the introduction of multiple NDs.

Conclusions

In this study, we presented an advanced PIDON framework designed to address the challenges posed by highly nonlinear and complex physical systems, specifically in the context of composites autoclave processing. Our approach integrates nonlinear decoders, domain decomposition, and curriculum learning strategies, which together notably improve the model's ability to capture complex solution operators and provide accurate predictions and generalizability across a wide range of input functions and design spaces. The introduced enhancements effectively addressed the shortcomings of vanilla PIDON architecture, resulting in a robust and reliable predictive performance. The advanced PIDON's zero-shot and real time inference capabilities make it highly suitable for applications in digital twins and Industry 4.0, where real-time data and simulations are crucial for monitoring and controlling manufacturing processes. This ability to provide swift and precise insights ensures that PIDON can play a pivotal role in enhancing the efficiency and reliability of composites manufacturing and other related fields.

References

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Boullé, N. and Townsend, A. A mathematical guide to operator learning. *arXiv preprint arXiv:2312.14688*, 2023.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Chen, G., Li, Y., Meng, Q., Zhou, J., Hao, X., et al. Residual fourier neural operator for thermochemical curing of composites. *arXiv preprint arXiv:2111.10262*, 2021.

Chen, G., Li, Y., Liu, X., Mehdi-Souzani, C., Meng, Q., Zhou, J., and Hao, X. Physics-guided neural operator for data-driven composites manufacturing process modelling. *Journal of Manufacturing Systems*, 70:217–229, 2023.

Chen, T. and Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Fabris, J. N. *A framework for formalizing science based composites manufacturing practice*. PhD thesis, University of British Columbia, 2018.

Fang, Z., Wang, S., and Perdikaris, P. Learning only on boundaries: A physics-informed neural operator for solving parametric partial differential equations in complex geometries. *Neural Computation*, 36(3):475–498, 2024.

Gao, H., Sun, L., and Wang, J.-X. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428: 110079, 2021.

Gin, C. R., Shea, D. E., Brunton, S. L., and Kutz, J. N. Deepgreen: deep learning of green's functions for nonlinear boundary value problems. *Scientific reports*, 11(1):21614, 2021.

Haghighat, E., bin Waheed, U., and Karniadakis, G. En-deeponet: An enrichment approach for enhancing the expressivity of neural operators with applications to seismology. *Computer Methods in Applied Mechanics and Engineering*, 420:116681, 2024.

Hubert, P., Johnston, A., Poursartip, A., and Nelson, K. Cure kinetics and viscosity models for hexcel 8552 epoxy resin. In *International SAMPE symposium and exhibition*, pp. 2341–2354. SAMPE; 1999, 2001.

Jagtap, A. D. and Karniadakis, G. E. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5), 2020.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

Kumar, V., Goswami, S., Smith, D. J., and Karniadakis, G. E. Real-time prediction of gas flow dynamics in diesel engines using a deep neural operator framework. *arXiv preprint arXiv:2304.00567*, 2023.

Lanthaler, S., Mishra, S., and Karniadakis, G. E. Error estimates for deeponets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022.

Lee, J. Y., Cho, S. W., and Hwang, H. J. Hyperdeeponet: learning operator with complex target function space using the limited resources via hypernetwork. *arXiv preprint arXiv:2312.15949*, 2023.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., and Karniadakis, G. E. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

Lucia, D. J., Beran, P. S., and Silva, W. A. Reduced-order modeling: new approaches for computational physics. *Progress in aerospace sciences*, 40(1-2):51–117, 2004.

Majumdar, R., Jadhav, V., Deodhar, A., Karande, S., Vig, L., and Runkana, V. Hxpinn: A hypernetwork-based physics-informed neural network for real-time monitoring of an industrial heat exchanger. *Numerical Heat Transfer, Part B: Fundamentals*, pp. 1–22, 2024.

Mattey, R. and Ghosh, S. A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, 2022.

Meng, Q., Li, Y., Liu, X., Chen, G., and Hao, X. A novel physics-informed neural operator for thermochemical curing analysis of carbon-fibre-reinforced thermosetting composites. *Composite Structures*, 321:117197, 2023.

Niaki, S. A., Haghighat, E., Campbell, T., Poursartip, A., and Vaziri, R. Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture. *Computer Methods in Applied Mechanics and Engineering*, 384:113959, 2021.

Rahman, M. A., George, R. J., Elleithy, M., Leibovici, D., Li, Z., Bonev, B., White, C., Berner, J., Yeh, R. A., Kossaifi, J., et al. Pretraining codomain attention neural operators for solving multiphysics pdes. *arXiv preprint arXiv:2403.12553*, 2024.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Ramezankhani, M. and Milani, A. S. A sequential meta-transfer (smt) learning to combat complexities of physics-informed neural networks: Application to composites autoclave processing. *arXiv preprint arXiv:2308.06447*, 2023.

Rashid, M. M., Pittie, T., Chakraborty, S., and Krishnan, N. A. Learning the stress-strain fields in digital composites using fourier neural operator. *Iscience*, 25(11), 2022.

Seidman, J., Kissas, G., Perdikaris, P., and Pappas, G. J. Nomad: Nonlinear manifold decoders for operator learning. *Advances in Neural Information Processing Systems*, 35: 5601–5613, 2022.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Strong, A. B. *Fundamentals of composites manufacturing: materials, methods and applications*. Society of manufacturing engineers, 2008.

Tripura, T. and Chakraborty, S. Wavelet neural operator: a neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*, 2022.

Van Ee, D. and Poursartip, A. Hexply 8552 material properties database for use with compro cca and raven. *Version 0.9. NCAMP. Wichita, KS*, 2009.

Wang, S., Wang, H., and Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40): eabi8605, 2021.

Wang, S., Yu, X., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Wang, S., Sankaran, S., and Perdikaris, P. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, 2024.

Wight, C. L. and Zhao, J. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.

Zauderer, E. *Partial differential equations of applied mathematics*. John Wiley & Sons, 2011.

A. Composites autoclave processing case study

A.1. Governing equations

The one-dimensional thermochemical behavior of a composite-tool system in an autoclave is governed by an anisotropic heat conduction equation with an internal heat generation term $\dot{Q} = b_c \frac{\partial \alpha}{\partial t}$ accounting for the exothermic chemical reaction of the resin matrix during the curing process:

$$\begin{cases} \dfrac{\partial T_t}{\partial t} = a_t \dfrac{\partial^2 T_t}{\partial z^2} & z \in [0, L_1] \\ \dfrac{\partial T_c}{\partial t} = a_c \dfrac{\partial^2 T_c}{\partial z^2} + b_c \dfrac{\partial \alpha}{\partial t} & z \in [L_1, L_2] \end{cases} \quad \text{where } a = \frac{k}{\rho C_p} \text{ and } b = \frac{v_r \rho_r H_r}{\rho C_p}. \tag{5}$$

where, T is the temperature α is the DoC, L is the material length, and t and z are the spatiotemporal coordinates. Subscripts t, c, and r, represent the tool, composite part and resin, respectively. a denotes the thermal diffusivity, b is the heat generation coefficient, and k, p and C_p are the thermal conductivity, density and specific heat capacity. v and H represent the volume fraction and heat of reaction per unit mass. In the curing process of a composite system with thermoset resin, the cure rate $\frac{\partial \alpha}{\partial t}$ is determined by the resin's cure kinetics and is typically described by an ordinary differential equation. For the 8552 epoxy resin system, used in this study, the cure kinetics have been previously developed (Hubert et al., 2001), and can be expressed as follows:

$$\frac{\partial \alpha}{\partial t} = \frac{A \exp(-\frac{\Delta E}{RT}))}{1 + \exp(C(\alpha - (C_0 + C_T T)))} \alpha^m (1 - \alpha)^n. \tag{6}$$

Here, ΔE represents the activation energy, R is the gas constant, and C_0, C_T, m, n and A are experimentally determined constants. Table 3 provides a summary of the parameter values used in the cure kinetics equations for this study.

Table 3. Summary of parameters used in heat transfer and cure kinetics governing equations.

PARAMETER	DESCRIPTION	VALUE
ΔE	ACTIVATION ENERGY	66.5 (kJ/gmol)
R	GAS CONSTANT	8.314
A	PRE-EXPONENTIAL CURE RATE COEFFICIENT	1.53×10^5 (1/s)
m	FIRST EXPONENTIAL CONSTANT	0.813
n	SECOND EXPONENTIAL CONSTANT	2.74
C	DIFFUSION CONSTANT	43.1
C_0	CRITICAL DEGREE OF CURE AT $T = 0$ K	-1.684
C_T	CRITICAL RESIN DEGREE OF CURE CONSTANT	5.475×10^{-3} (1/K)

The initial conditions of the coupled system described above can be specified as:

$$\begin{aligned} T_c \mid_{t=0} &= T_0(x) \\ T_t \mid_{t=0} &= T_0(x) \\ \alpha \mid_{t=0} &= \alpha_0(x). \end{aligned} \tag{7}$$

T_0 represents the part's initial temperature, which is typically considered uniform throughout. In this study, the initial temperature is assumed to be 20°C. α_0 denotes the initial DoC of the resin system, and for an uncured part, it is assumed to be zero or a very small value; in this study, a value of 0.05 is used. Considering the convective heat transfer between the autoclave air T_a and the composites system, the boundary conditions are governed by:

$$\begin{aligned} (T_a - T_c \mid_{z=L_2}) &= \frac{k_c}{h_{top}} \frac{\partial T_c}{\partial z} \mid_{z=L_2} \\ (T_t \mid_{z=0} - T_a) &= \frac{k_t}{h_{bot}} \frac{\partial T_t}{\partial z} \mid_{z=0} \end{aligned} \tag{8}$$

where h_{top} and h_{bot} are the HTCs on the top and bottom surfaces of the composite-tool system. Furthermore, the solution of the described system must satisfy the following continuity conditions between the part and the tool:

$$\begin{aligned} k_t \frac{\partial T_t}{\partial z} \mid_{z=L_1^-} &= k_c \frac{\partial T_c}{\partial z} \mid_{z=L_1^+} \\ T_t \mid_{z=L_1^-} &= T_c \mid_{z=L_1^+} . \end{aligned} \tag{9}$$

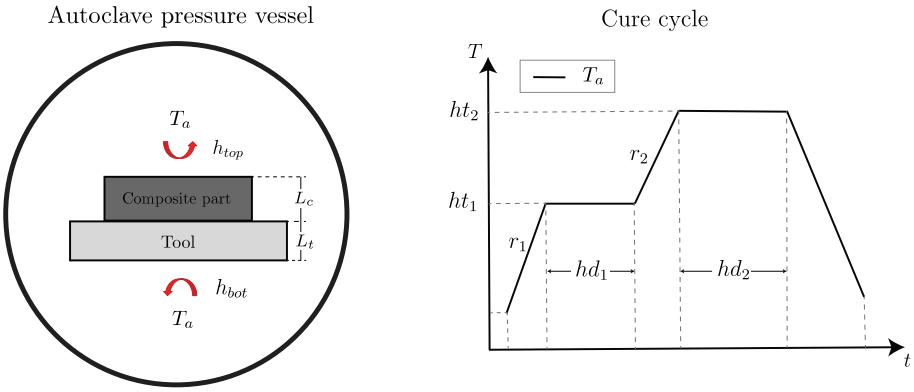

Figure 6. Schematic of Autoclave pressure vessel and typical two-hold cure cycle recipe for autoclave air temperature T_a in composites manufacturing.

A.2. Spatial local coordinates

Figure 1 illustrates the local coordinates x_1 and x_2, which are defined to manage inconsistencies in the total length of the system and the interface location resulting from the varying part and tool thicknesses. Substituting the local coordinates into the heat conduction, heat convection, and continuity equations results in the following transformed governing equations:

$$\begin{cases} \dfrac{\partial T_t}{\partial t} = \dfrac{a_t}{L_t^2} \dfrac{\partial^2 T_t}{\partial x_1^2} & x_1 \in [0,1] \\ \dfrac{\partial T_c}{\partial t} = \dfrac{a_c}{L_c^2} \dfrac{\partial^2 T_c}{\partial x_2^2} + b_c \dfrac{\partial \alpha}{\partial t} & x_2 \in [0,1] \end{cases} \tag{10}$$

$$\begin{aligned} \frac{\partial T_c}{\partial x_2} \mid_{x_2=1} &= \frac{h_{top} L_c}{k_c} (T_a - T_c \mid_{x_2=1}) \\ \frac{\partial T_t}{\partial x_1} \mid_{x_1=0} &= \frac{h_{bot} L_t}{k_t} (T_t \mid_{x_1=0} - T_a) \\ T_t \mid_{x_1=1} &= T_c \mid_{x_2=0} \\ \frac{k_t}{L_t} \frac{\partial T_t}{\partial x_1} \mid_{x_1=1} &= \frac{k_c}{L_c} \frac{\partial T_c}{\partial x_2} \mid_{x_2=0} . \end{aligned} \tag{11}$$

The use of local coordinates ensures identical coordinate domain size across all bi-material systems selected for training and testing. The thickness variation is then appropriately accounted for within the parameters of the differential equations (i.e., via the presence of L_c and L_t in such equations). This also enables treating the composite part and tool as standalone systems trained on separate networks with input variables normalized to 0 and 1. Specifically, as discussed in Section 2.4, two DeepONets are allocated to capture the thermal behaviors of the part and tooling separately. While the individual DeepONets are responsible for learning separate systems, they also need to simultaneously satisfy the continuity conditions

at the interface of the two materials. This is accomplished by defining two additional loss terms:

$$\mathcal{L}_{CT_1}(\theta_t,\theta_c) = \frac{1}{NQ_{ct}}\sum_{i=1}^{N}\sum_{j=1}^{Q_{ct}} |G^{T_t}_{\theta_t}(u^{(i)})(y^{(i)}_j)) - G^{T_c}_{\theta_c}(u^{(i)})(y^{(i)}_j))|^2$$
$$\mathcal{L}_{CT_2}(\theta_t,\theta_c) = \frac{1}{NQ_{ct}}\sum_{i=1}^{N}\sum_{j=1}^{Q_{ct}} |\frac{k_t}{L_t}\frac{\partial G^{T_t}_{\theta_t}(u^{(i)})(y^{(i)}_j))}{\partial x_1} - \frac{k_c}{L_c}\frac{\partial G^{T_c}_{\theta_c}(u^{(i)})(y^{(i)}_j))}{\partial x_2}|^2. \tag{12}$$

Here θ_t and θ_c denote the weights associated with the part DeepONet G^{T_t} and tool DeepONet G^{T_c}, respectively (see Figure 3). Q_{ct} is the number of residual points evaluated at the materials' interface.

A.3. Training procedure

During the training phase, G^{T_t} is responsible for enforcing the tool's IC, bottom surface BC, and tool's PDE. Similarly, G^{T_c} ensures the part's IC, top surface BC, and the part's PDE are satisfied. Additionally, the continuity conditions between the tool and part are maintained by jointly updating the weights of G^{T_t} and G^{T_c}. Similarly, ODE loss (resin cure kinetics) is minimized by the sequential co-training of G^{T_c} and G^{α}. Furthermore, G^{α} is tasked with minimizing the initial condition loss associated with DoC. We implemented the sequential learning approach (Niaki et al., 2021) to train the operators. The training began with updating the weights of G^{T_c} and G^{T_t} through their associated loss terms for 10 epochs while keeping the G^{α}'s weights constant. Subsequently, G^{α} was trained for 10 epochs while the other two operators remained frozen. The training ends after repeating this procedure 10 times.

The input of BN2 is the sensory information of the air profile (i.e., cure cycle) surrounding the composite system during the curing process which enforces the boundary conditions. Various cure cycles with different numbers of isothermal holds as well as different heat ramp rates and hold durations are considered. For each cure cycle, the air temperature is recorded at 100 specified time steps (i.e., sensor locations) and the data is passed to BN2. BN1 on the other hand is fed with the remaining time-invariant process parameters. In this study, four process parameters including the top HTC, bottom HTC, tool's thickness, and composite part's thickness are considered (Figure 7). Table 4 summarizes the design parameters and their corresponding ranges used for training the PIDON models in this study.

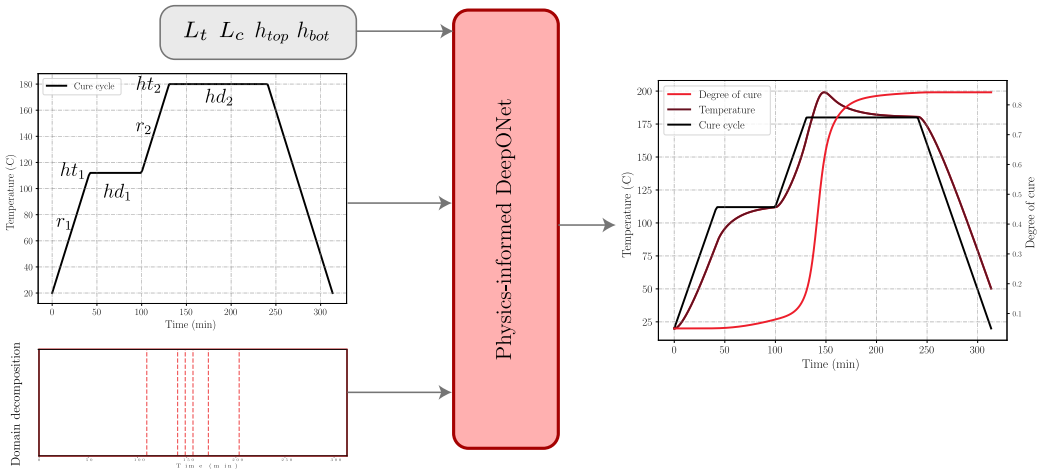

Figure 7. Proposed PIDON input-output functions mapping. PIDON takes in time-dependent (cure cycle) and time-independent $(L_t, L_c, h_{top}, h_{bot})$ functions as input and outputs temperature and DoC fields. The arrangement of temporal subdomains is also presented.

B. DeepONet

Based on the universal approximation theorem for operators, (Chen & Chen, 1995) proposed *operator nets*, a neural network architecture that approximates nonlinear operators that map infinite dimensional Banach spaces. An operator net consists of

Table 4. Design variables (input functions) and their corresponding ranges for three different design space sizes.

DESIGN PARAMETER	DESCRIPTION	DESIGN SPACE SIZE		
		SMALL	MEDIUM	LARGE
h_{top} (W/m^2 K)	TOP HTC	[90, 120]	[80, 120]	[70, 120]
h_{bot} (W/m^2 K)	BOTTOM HTC	[60, 90]	[50, 90]	[50, 100]
r_1 (°C/min)	RAMP 1	[1.9, 2.8]	[1.7, 3]	[1.5, 3]
ht_1 (°C)	HOLD 1 TEMPERATURE	[110, 115]	[105, 115]	[105, 120]
hd_1 (min)	HOLD 1 DURATION	[55, 63]	[52, 63]	[50, 65]
r_2 (°C/min)	RAMP 2	[1.9, 2.8]	[1.7, 3]	[1.5, 3]
ht_2 (°C)	HOLD 2 TEMPERATURE	[178, 183]	[175, 185]	[170, 185]
hd_2 (min)	HOLD 2 DURATION	[105, 115]	[105, 120]	[105, 120]
L_t (cm)	TOOL THICKNESS	[2, 3.5]	[2, 4]	[2, 5]
L_c (cm)	PART THICKNESS	[2.5, 3.5]	[2.5, 3.5]	[2.5, 3.5]

two shallow neural networks, namely, branch net and trunk net, which encode the input functions and system coordinates, respectively. The branch and trunk nets are merged to approximate the underlying operator solution. (Lu et al., 2021) proposed a more expressive variant of the operator net named DeepONet by replacing shallow networks with deep neural networks. The architecture of the DeepONet can naturally be decomposed into three main components: an encoder, an approximator, and a decoder, as illustrated in Figure 8.a (Lanthaler et al., 2022). The encoder is responsible for mapping the infinite-dimensional input space to a finite-dimensional space. This is crucial for training the operator as the input functions must be expressed discretely to implement the network approximations. In other words, the continuous input functions are mapped to their discretized representation (i.e., finite-dimensional space) by pointwise evaluations at m fixed sensor points x_j. The approximator is parameterized by a deep neural network that maps the sensor point evaluations $u = [u(x_1), u(x_2), ..., u(x_m)]$ to a finite-dimensional feature representation $b = [b_1, b_2, ..., b_q]^T \in R^q$. The composition of the encoder and approximator results in the branch net of DeepONet expressed by $\beta(u) = \mathcal{A} \circ \mathcal{E}(u)$. Similar to the branch net, the trunk net is parameterized by a deep neural network that encodes the inputs of the PDE system y to a feature embedding $t = [t_1, t_2, ..., t_q]^T \in R^q$ with the same size as the branch net's output (Figure 8.b). Finally, the decoder takes the output of the branch (q coefficients) and trunk nets (q basis functions) and calculates the DeepONet's output using an element-wise product operation followed by a summation, $G_\theta(u)(y) = \sum_{k=1}^{q} b_k t_k + b_0$. The bias term b_0 is added in practice to improve the generalization performance of DeepONet G. The decoder can also be seen as a single network (trunk net) with its weights in the last layer parameterized by another network (branch net). In a supervised learning fashion, the DeepONet can be trained by minimizing the error between the model's predicted output and the actual operator solution across a range of training input functions.

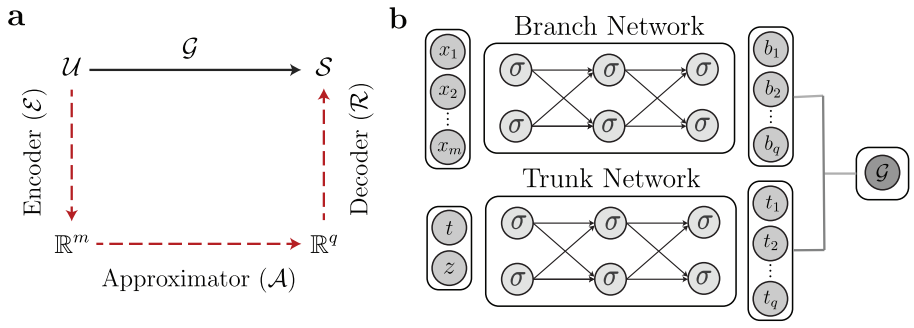

Figure 8. Schematic of DeepONet decomposition into Encoder, Approximator, and Decoder (a); architecture of vanilla DeepONet with a branch net, a trunk net, and a linear decoder.

C. Additional results

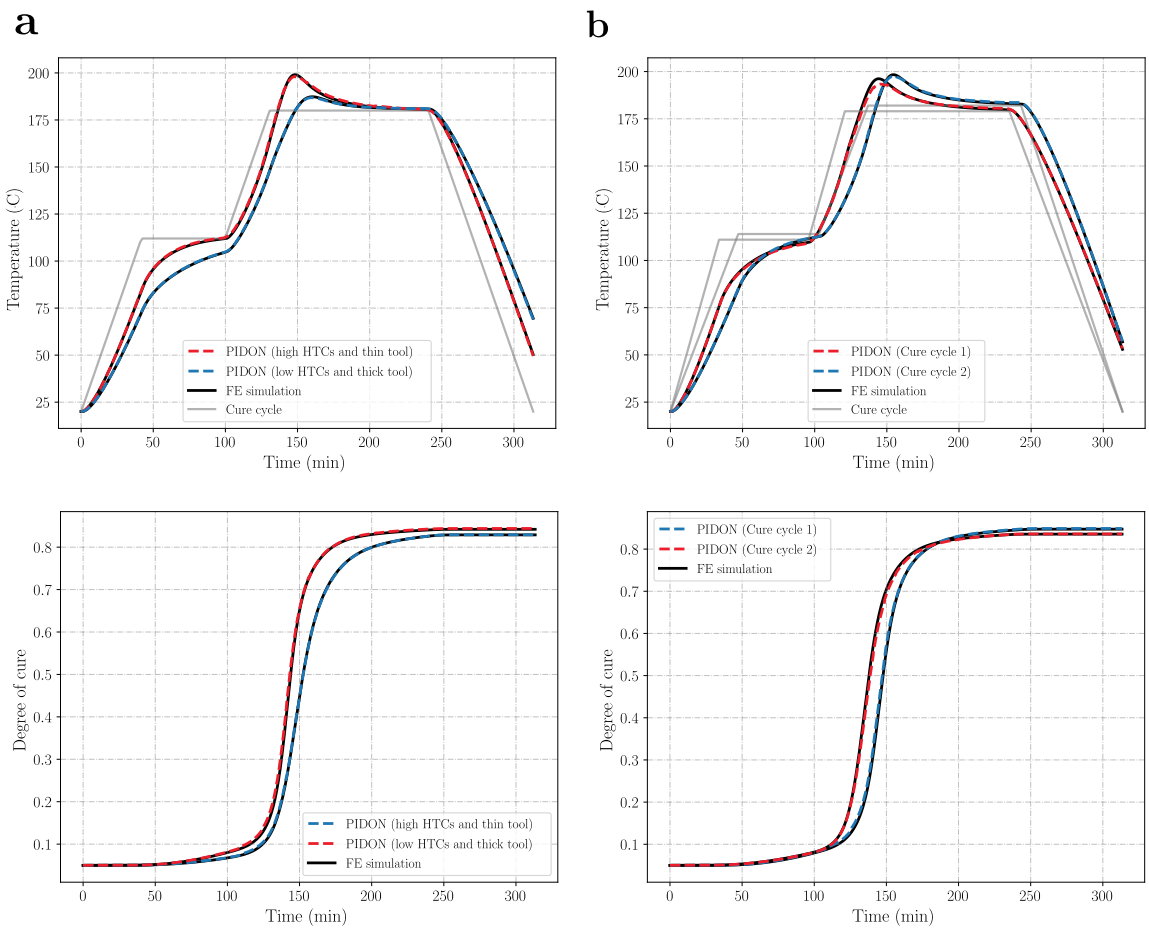

Figure 9. Zero-shot prediction performance of PIDON at composite part mid-point for: a) high HTCs and thin tool vs. low HTCs and thick tool; b) two different cure cycles (shown in gray.)

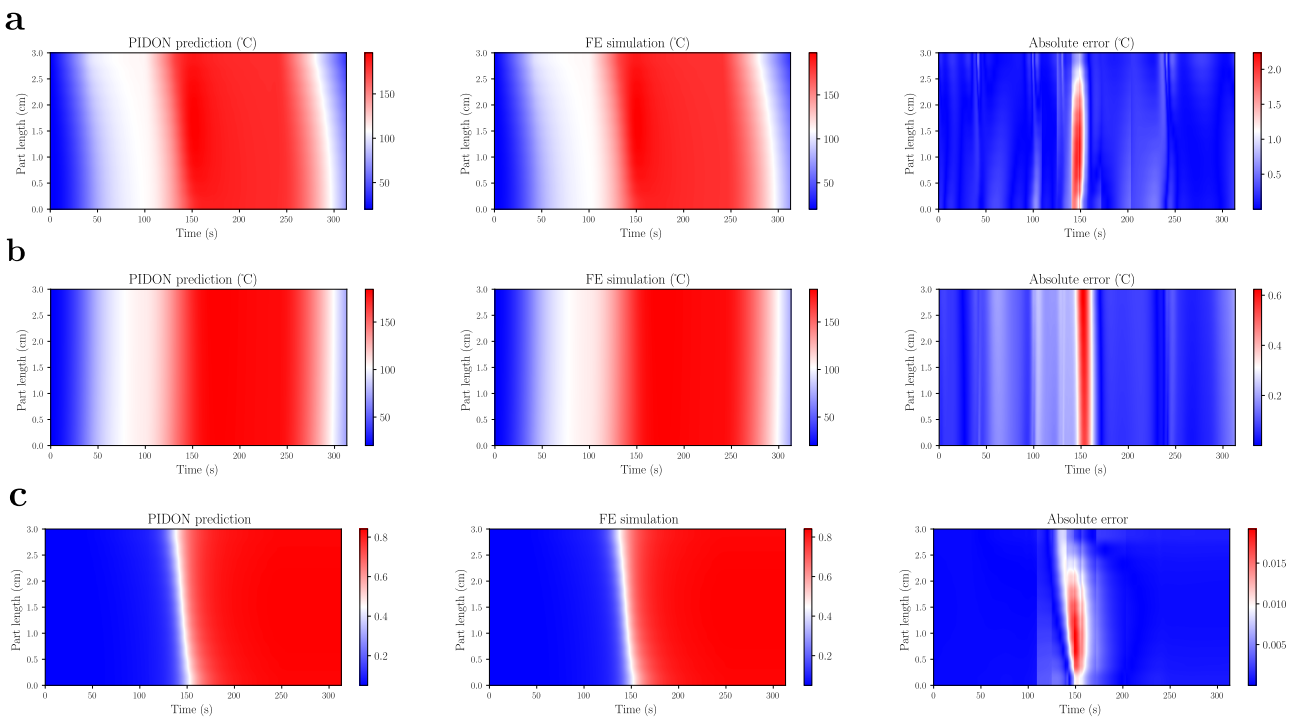

Figure 10. Comparison of PIDON prediction and FE simulation for Part temperature (a), tool temperature (b) and DoC (c) for a test case with the design variables: $h_{top} = 75, h_{bot} = 115, r_1 = r_2 = 2.2, ht_1 = 110, hd_1 = 58, ht_2 = 180, hd_2 = 105, L_t = 0.025, L_c = 0.03$.

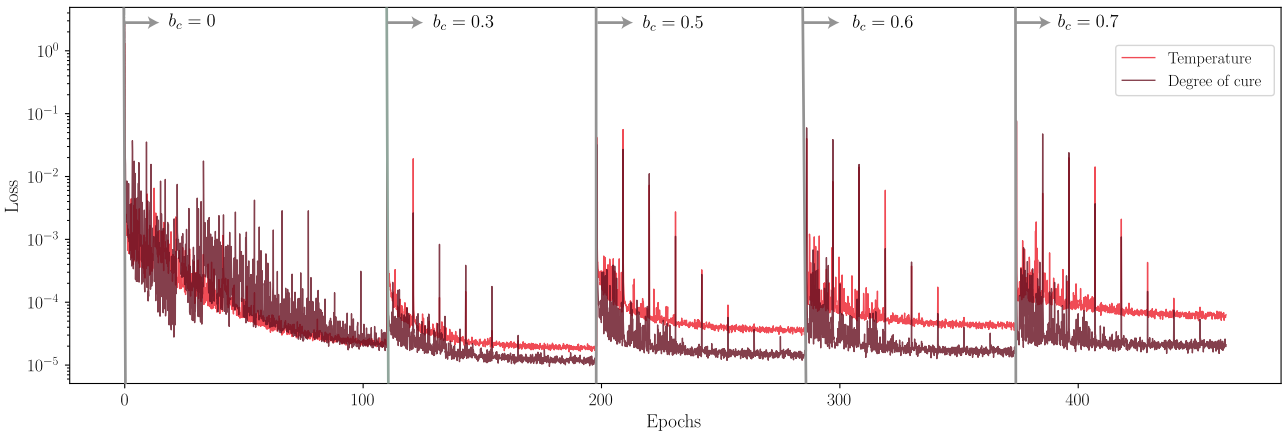

Figure 11. The evolution of temperature and DoC training losses during curriculum learning at various values of heat generation coefficient b_c.