Solid Freeform Fabrication 2024: Proceedings of the 35th Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference

Toward parametric heat transfer solvers in additive manufacturing

Akshay. J. Thomas*, Eduardo Barocio*, Ilias Bilionis*, and R. Byron Pipes**

*School of Mechanical Engineering - Purdue University. West Lafayette, IN, USA.
** School of Aeronautics and Astronautics - Purdue University. West Lafayette, IN, USA.

Abstract

Physics informed neural networks (PINNs) have recently been a popular framework to integrate experimental data and physics-based constraints specified via partial differential equations. However, the application of PINNs to additive manufacturing is limited due to the fact that a suitable physics-based loss function is missing for geometries that evolve. The objective of this work is to address this gap. We propose a loss function for PINNs to solve the transfer equation on evolving geometries without a mesh-based discretization. We use our methodology to predict the temperature evolution as a single bead is being deposited. For simplicity, 2D cases are the focus of the work. We consider various cases of mixed Dirichlet and Neumann boundary conditions and compare our results to finite element simulations. We also present guidelines to obtain consistent results using the proposed method. We finally discuss the potential of our method in solving parametric heat transfer problems in additive manufacturing.

Introduction

The increased availability of advanced computational resources, especially graphical processing units (GPUs), has led to the use of machine learning algorithms in solving applied and classical problems in science and engineering. Physics-informed neural networks [1] are a technique to seamlessly combine observational data and known physics in the form of partial differential equations (PDEs). This is achieved by parameterizing a quantity of interest (for example, a temperature field, a displacement field, etc.) using deep neural networks and solving a multi-task learning problem to match observational data while respecting the known physics that governs the physical process. There has also been success in PINNs being used as solvers in the absence of observational data [2]. PINNs have shown promising results in various fields of computational science and engineering, including solid mechanics [3], [4], [5], [6], fluid mechanics [7], [8], and material science [9], [10].

PINNs have also been explored in manufacturing applications. Niacki et al. [11] explored the application of PINNs to model the thermochemical curing process of composite-tool systems. The authors solve a coupled set of differential equations to predict the degree of cure in a composite part as it is manufactured. Another example of the application of PINNs is the work in [12]. They solve a parametric problem to optimize process conditions in composite manufacturing. [13] combined experimental data from a thermal camera with known physics to predict full-field temperature history and to discover unknown material and process parameters. The work in [13] used PINNs to predict the thermal history in metal additive manufacturing without simulation data. Note, however, that the domain itself was not evolving in this case, and parametric solutions were obtained via a transfer learning approach. The work in [14] used physical laws to augment temperature data obtained from sensors during a directed energy deposition process. PINNs were also used to predict grain structure in directed energy deposition[15], process parameter estimation in polymer-based laser sintering[16], and to improve a thermal model in stereolithography [17]. In the research mentioned above, PINNs do not track the evolving geometries and boundary conditions. In the cases where the geometry evolves [14], data from sensors are used, and the physical laws are used to augment the observational data. Further, there is no fundamental research on how an analyst can build parametric solvers when the geometry evolves.

Therefore, our work focuses on advancing the start of art in PINNs by proposing a new loss function that can tackle evolving geometries and automatically respect the causality of the problem. The main advantages of using PINNs are:

(i) *Attaining infinitesimal temporal resolution* – PINNs do not require spatial mesh or time incrementation schemes. This is a considerable advantage when small time increments are necessary.

(ii) *PINNs do not necessarily require observational data* – In the absence of observational data, PINNs act as PDE solvers. When observational data from experiments are available, one can augment PINNs by adding a loss term to the existing framework.

(iii) *Exact gradient computations are available via automatic differentiation* - This is particularly useful when temperature gradients are required to estimate the quality of bonds formed in additive manufacturing.

Methodology

Overview of finite element modeling approach

We first describe the finite element (FE) method to solve PDEs on evolving geometries. This method is termed progressive element activation [18]or the quiet element method in literature[19]. The method begins by meshing the entire geometry to be deposited, followed by the specification of the G-code. At a given time instance, a module finds the part of the FE mesh that intersects with the G-code and assigns "mass" to these elements. From an implementation point of view, the stiffness matrix is assembled for the entire geometry, and the rows corresponding to the "inactive" nodes are assigned a minimal mass; therefore, their contribution to the solution is close to zero. For heat transfer simulation this is achieved by multiplying the true thermal conductivity and the specific heat by a very small value to minimize the transfer of energy to the inactive part of the mesh. This method can be more easily visualized in Fig. 1. This method involves two levels of discretization. The first is the spatial discretization induced by the finite elements describing the geometry. Second, a time incrementation scheme that governs the amount of material deposited at each increment. In cases where sharp temperature gradients are crucial, for example, when estimating bonding characteristics, very small increments need to be used. This can prove computationally expensive. In this scenario, a natural question arises: Can $\Delta T \rightarrow 0$. This is where we believe the paradigm of PINNs holds potential. We describe a general introduction to PINNs and our formulation in the following sections.

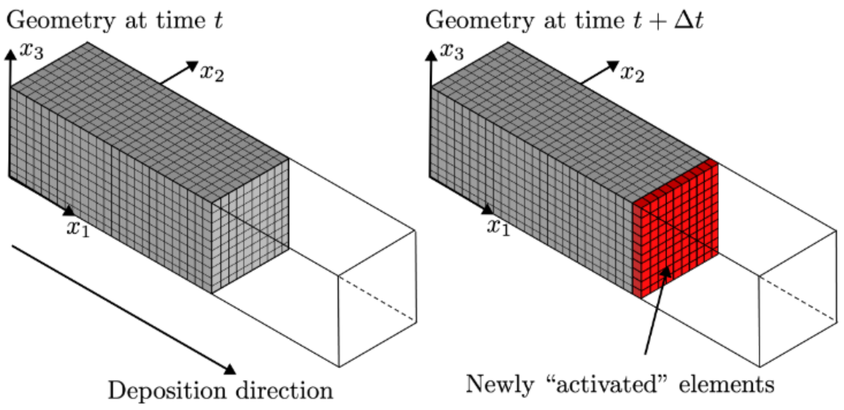


Figure 1: Illustration of the element activation technique used in finite element.

Physics-informed neural networks

First, we provide an overview of physics-informed neural networks (PINNs). Consider a physics object, denoted as $\mathcal{X}$, which is a subset of the d −dimensional Euclidean space. A physical process induces a field, u, on this object that evolves in time, t. Let the evolution of u be governed by the initial boundary value differential equation:

$$u_t + Au = s, \qquad x \in \mathcal{X}, t \in [0, T], \tag{1}$$

where u_t denotes the time derivative of u, and A is a differential operator. The term s represents the source term which can be constant or a function of space, time, or the field u. The boundary and initial conditions for this problem are given by

$$Bu = 0 \;\; x \in \mathrm{T}_D \; t \in [0, T], \tag{2}$$

and

$$u(0, x) = h(x) x \in \mathcal{X}, \tag{3}$$

respectively. When an analytical solution is unavailable, we aim to find an approximate solution that satisfies the initial boundary value problem. We represent the approximate solution using a neural network and denote it as u_θ. The variable θ denotes the parameters of the neural network. By plugging u_θ into the governing equations, we obtain the following residual, boundary, and initial condition residuals:

$$r_\theta^p(t, x) = \frac{\partial u_\theta}{\partial t}(t, x) + A\, u_\theta(t, x) - s\,, \tag{4}$$

$$r_\theta^{bc}(t, x) = Bu_\theta(t, x), \tag{5}$$

and

$$r_\theta^{ic}(t, x) = u_\theta(0, x) - h(x). \tag{6}$$

We can estimate θ, by minimizing a weighted composite loss function:

$$l(\theta) = \frac{\lambda_p}{N_p} \sum_{i=1}^{N_P} [r_\theta^p(t_i, x_i)]^2 \; + \frac{\lambda_{bc}}{N_{bc}} \sum_{j=1}^{N_{bc}} [r_\theta^{bc}(t_j, x_j)]^2 + \; + \frac{\lambda_{ic}}{N_{ic}} \sum_{k=1}^{N_{ic}} [r_\theta^{ic}(0, x_k)]^2\,. \tag{7}$$

This composite loss can be minimized using a variant of stochastic gradient descent, with the ADAM [20]optimizer being the most commonly used. Automatic differentiation makes the gradients exactly available for the higher gradient calculations.

<u>Loss to tackle evolving geometries</u>

For additive manufacturing applications where the material is deposited in a predefined fashion, we propose the loss function given by

$$l_e(\theta) = \int dt \;\; l(t, x(t); \theta). \tag{8}$$

Note the similarities between the loss in Eq. (8) and the one in [21]. The main difference is that we do not need an extra hyperparameter that controls the degree of respecting causality. To proceed, we need to convert the integral statement in Eq. (8) to a quantity that is amenable to optimization methods. This is achieved by finding an unbiased estimator as

$$\widehat{l_e}(\theta) = \frac{|T|}{N_m} \sum_{m=1}^{N_m} l(t_m, x(t_m); \theta)\,. \tag{9}$$

The unbiased estimator ensures the stochastic gradient descent algorithm converges to a local minimum as long the learning rate is a converging sequence. The term $l(t_m, x(t_m); \theta)$ is some loss function that represents the residual of the PDE system, and$|T|$ is the total time. Here, l (the term inside the summation in Eq. 9) is chosen to be the squared loss common in literature. By working out the algebra, we arrive at a loss function defined as

$$\widehat{l_e}(\theta) = \frac{\lambda_p}{N_m N_p} \sum_{m=1}^{N_m} \sum_{i=1}^{N_P} [r_\theta^p(t_i, x_i(t_m))]^2 + \frac{\lambda_{bc}}{N_m N_{bc}} \sum_{m=1}^{N_m} \sum_{j=1}^{N_{bc}} [r_\theta^{bc}(t_j, x_j(t_m))]^2 + \frac{\lambda_{ic}}{N_m N_{ic}} \sum_{m=1}^{N_m} \sum_{k=1}^{N_{ic}} [r_\theta^{bc}(t_k, x_k(t_m))]^2 . \tag{10}$$

In the above equation, the terms on the right-hand side of the equality are arranged as the PDE residual, boundary condition residual, and initial condition residual, respectively. The loss function in Eq. (10) is similar to the loss functions used in literature [22]. However, the main difference is that a nested summation accounts for the evolution of geometry. We would also like to clarify that the third term in Eq. (10) does not necessarily mean that the initial condition is evolving. This term is only used when new material is added at a specified condition. In a heat transfer analysis of additive manufacturing, this condition corresponds to the material added at a specified temperature. If one wishes to specify an amount of material specified at a temperature at the time, *t=0*, one can specify this by adding a loss term to Eq. (10), specifying this condition. In the numerical examples, we study cases where certain boundaries do not evolve.

Implementation considerations

Below are a few implementation considerations that help with the converge of PINNs to the desired solution. The suggestions have been meticulously documented in [22]. We perform ablation tests based on these recent advancements to study the effect of each modification on the accuracy of solving PDEs in evolving geometries.

PDE non-dimensionalization

In classical machine learning, data normalization is an essential step before training to ensure that the inputs and outputs are on similar scales [23]. However, the range of target variables is not apriori in forward problems. Therefore, it is essential to ensure that the features and the output are within reasonable scales. One way to achieve this is by non-dimensionalizing the PDE. We scale the output to be on the order of one and also scale the input co-ordinates between zero and one. By this scaling, we end up with effective properties that are scaled versions of the original values. This ensures that the input variables are consistent with network initialization schemes, like the Glorot initialization scheme. Further, if the output and inputs vary significantly in scales, it can lead to difficulties in the training process. Non-dimensionalization alleviates this issue, facilitating convergence and learning meaningful correlations.

Architecture

Throughout our work, we approximate the latent function u using a deep neural network (DNN). DNNs are a powerful class of function approximators (a more detailed explanation can be found in [24]). Further, Wang et al. [24] also proposed a new architecture called the modified DNN that was shown to accelerate convergence and easier learning of sharp gradients. DNNs are also susceptible to spectral bias, which causes the learning of low-frequency solutions first. This hinders the learning of steep gradients and fine features of solutions, as described in [24]. To overcome the spectral bias of DNNs, Tanick et al. [26] proposed passing the inputs through a high-frequency mapping before being inputted to the DNN layers. This mapping, $\gamma(x)$ is

$$\gamma(x) = \begin{bmatrix} \cos(\boldsymbol{B}x) \\ \sin(\boldsymbol{B}x) \end{bmatrix}, \tag{11}$$

where $\boldsymbol{B} \in R^{m \times d}$ is sampled from a normal distribution with varianceσ^2. The variable d is the input dimension of the DNN, and the variable m is the number of frequency components the input is mapped to. In this paper, we choose $m = 128$ and $\sigma = 1$.

Training

A significant challenge in training PINNs is balancing the interplay between the various loss terms to ensure that one loss term does not dominate the others. Consider the loss term similar to the one derived earlier

$$\widehat{l_e}(\theta) = \lambda_p \widehat{l_p}(\theta) + \lambda_{bc} \widehat{l_{bc}}(\theta) + \lambda_{ic} \widehat{l_{ic}}(\theta). \tag{12}$$

To ensure the gradients of the three terms are equal, consider the updates to λ_p, λ_{bc}, and λ_{ic} as follows

$$\lambda_p^{update} = \frac{\|\nabla_\theta l_p(\theta)\| + \|\nabla_\theta l_{bc}(\theta)\| + \|\nabla_\theta l_{ic}(\theta)\|}{\|\nabla_\theta l_p(\theta)\|}, \tag{13}$$

$$\lambda_{bc}^{update} = \frac{\|\nabla_\theta l_p(\theta)\| + \|\nabla_\theta l_{bc}(\theta)\| + \|\nabla_\theta l_{ic}(\theta)\|}{\|\nabla_\theta l_{bc}(\theta)\|}, \tag{14}$$

$$\lambda_{ic}^{update} = \frac{\|\nabla_\theta l_p(\theta)\| + \|\nabla_\theta l_{bc}(\theta)\| + \|\nabla_\theta l_{ic}(\theta)\|}{\|\nabla_\theta l_{ic}(\theta)\|}, \tag{15}$$

where $\|\cdot\|$ is the L^2 norm. The new weights are then calculated by

$$\lambda_{new} = \alpha \lambda^{old} + (1 - \alpha)\lambda^{new}. \tag{16}$$

The variable α controls the trade-off between the updates and the old values. We observed that specifying an α value of 0.9 and updating every 1000 iterations (including the first iteration) provided consistent results.

For the optimizer, we observed consistent results using the ADAM optimizer with the default parameters suggested in [20]. The sampling of collocation points also play a crucial role in the convergence of the neural networks to the correct solution of the PDE. We observed that random sampling provided satisfactory results in our test cases. Further, we suggest random sampling as the first choice since we perform nested sampling. The sampling strategy involves sampling a random time and by sequentially sampling points in space based on the current state of the geometry. Our sampling strategy resembles the one presented in [27]. The nested summation in our loss function implies that as the geometry evolves in time, the collocation point density in space reduces. This forces the learning of the solution at earlier times first, without the need for either time marching or curriculum strategies. In our numerical experiments, we did not encounter casualty violations. Nonetheless, it would be interesting to see if other PDE experience causality violations.

Numerical Examples and Results

We apply our methodology to solve the heat transfer equation on an evolving geometry. We compare the PINN predictions to a validated FE solver developed at the composites manufacturing and simulation center[18]. To quantify the accuracy of our method, we use the relative L^2 error defined as

$$relative\ L^2\ error = \frac{\|u - u_\theta\|}{\|u\|}, \tag{17}$$

where u is the reference solution and u_θ is the PINN solution. We also evaluate the absolute error plots to quantify the accuracy of our predictions as the geometry evolves in time.

Example 1: Evolving geometry with evolving Dirichlet boundary condition

We will lay down the fundamental problem we investigate in this paper using this example. Consider a domain that evolves in time, $\Omega(t)$, such that $\Omega(0) = \emptyset$, and $\Omega(T) = \Omega'$, where T is the total time and Ω' is the final part geometry. The geometry evolves in the positive x direction at a constant velocity, $v_x = 1.0$ with no velocity component in the y-direction: $v = [v_x\ 0]$. The geometry evolves from $x_{min} = 0\ to\ x_{max} = 2.5$ in 2.5s. The dimension of the geometry in the y-direction is set to 1. The parabolic differential equation we aim to solve is

$$\alpha_x \frac{\partial^2 u}{\partial x^2} + \alpha_y \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t}\ on\ \Omega(t), \tag{18}$$

with the material being deposited at a value $u_I = 2.0$ on the boundary $\Gamma_I(t)$ which is mathematically written as

$$u = u_I\ on\ \Gamma_I(t), \tag{19}$$

and with a specified boundary condition of $\mathrm{u_B} = 0.5$ on the boundary $\Gamma_\mathrm{B}(\mathrm{t})$ as

$$u = u_B\ on\ \Gamma_B(t), \tag{20}$$

and an adiabatic boundary condition of

$$\mathrm{k_x}\frac{\partial \mathrm{u}}{\partial \mathrm{x}} = 0 \text{ at } \mathrm{x} = 0.$$

The variables α_x and α_y represent the thermal diffusivity in the x, and y direction, respectively. We choose values for the diffusivity as $\alpha_x = 1e-2$ and $\alpha_y = 6e-2$. The problem can be visualized in Fig. 2.

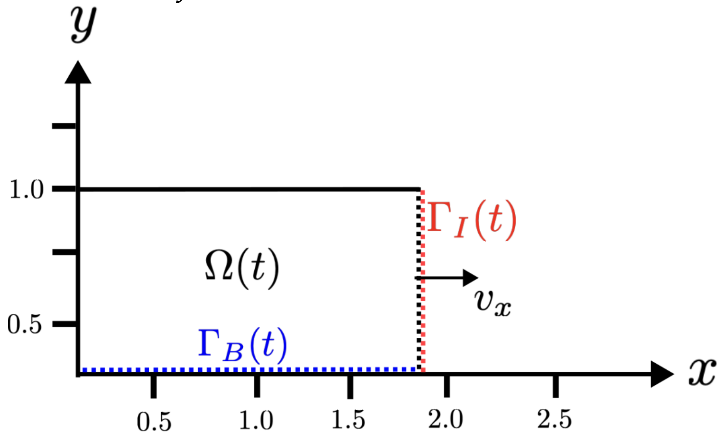


Figure 2: Illustration of example 1.

Before we proceed, we non-dimensionalize the PDE by introducing scaled versions of the fundamental variables as

$$\tilde{u} = \frac{u}{u_I}, \tilde{x} = \frac{x}{x_{max}}, \tilde{y} = \frac{y}{y_{max}}, \tilde{t} = \frac{t}{T}. \tag{21}$$

We substitute the above into the original PDE and solve the problem in the scaled domain. The loss term for this problem is

$$\hat{l}(\theta) = \lambda_p \widehat{l_p}(\theta) + \lambda_{bc}\widehat{l_{bc}}(\theta) + \lambda_{ic}\widehat{l_{ic}}(\theta) + \lambda_n \widehat{l_n}\ (\theta), \tag{22}$$

$$\hat{l}(\theta) = \frac{\lambda_p}{N_m N_p} \sum_{m=1}^{N_m} \sum_{i=1}^{N_P} \left[\tilde{\alpha}_x \frac{\partial^2 \tilde{u}_\theta}{\partial x^2} + \tilde{\alpha}_y \frac{\partial^2 \tilde{u}_\theta}{\partial \tilde{y}^2} - \frac{\partial \widetilde{u_\theta}}{\partial \tilde{t}} \right]^2 + \frac{\lambda_{bc}}{N_m N_{bc}} \sum_{m=1}^{N_m} \sum_{j=1}^{N_{bc}} \left[\tilde{u}_\theta - \frac{u_B}{u_I} \right]^2$$
$$+ \frac{\lambda_{ic}}{N_m N_{ic}} \sum_{m=1}^{N_m} \sum_{j=1}^{N_{ic}} \left[\tilde{u}_\theta - \frac{u_I}{u_I} \right]^2 + \frac{\lambda_n}{N_n} \sum_{n=1}^{N_n} \left[\tilde{\mathrm{k}}_{\mathrm{x}} \frac{\partial \tilde{u}_\theta}{\partial \tilde{x}} \right]^2. \tag{23}$$

Note that the first three terms on the right hand side have nested summations to account for the evolution of geometry. However, the last term does not require this nested summation since this boundary does not evolve. For this example, we use a DNN with 4 hidden layers and 128 neurons in each layer as the backbone model. We use the Tanh activation function. The parameters are initialized using the Glorot scheme, and the network parameters are trained using the ADAM optimizer with an initial learning rate of 5e-4, with an exponential decay at a rate of 0.9 every 1000 iterations. The total number of iterations is set to be 200000. The different batch sizes are chosen as $N_p = 8, N_p = N_b = N_{ic} = N_n = 1024$.

The L^2 error is calculated only for the geometry that has been deposited. We summarize the average L^2 error over the entire time for the ablation studies in Table 1.

Table 1: Summary of ablation studies performed on example 1

Ablation Settings			Evaluation Metric	
Fourier Features	Modified DNN	Gradient Norm	Average Rel. L2 error	Computational Time (min)
✓	✓	✓	6.4×10^{-3}	39.4
✓	✓	✗	6.9×10^{-3}	38.4
✓	✗	✓	7.0×10^{-3}	25.0
✓	✗	✗	6.6×10^{-3}	24.9
✗	✓	✓	8.0×10^{-3}	31.9
✗	✓	✗	6.6×10^{-3}	32.1
✗	✗	✓	7.3×10^{-3}	20.5
✗	✗	✗	7.0×10^{-3}	20.3

Note that all cases have low error, and we investigate the case with the lowest error. This is the case where the modified DNN is used with the Fourier features and loss gradient balancing. The evolution of the loss and the loss weights for this case is shown in Fig. 3. In Fig. 4, we plot the spatial temperature field at various time instances and observe that our PINN predictions are in close agreement with the FE solution.

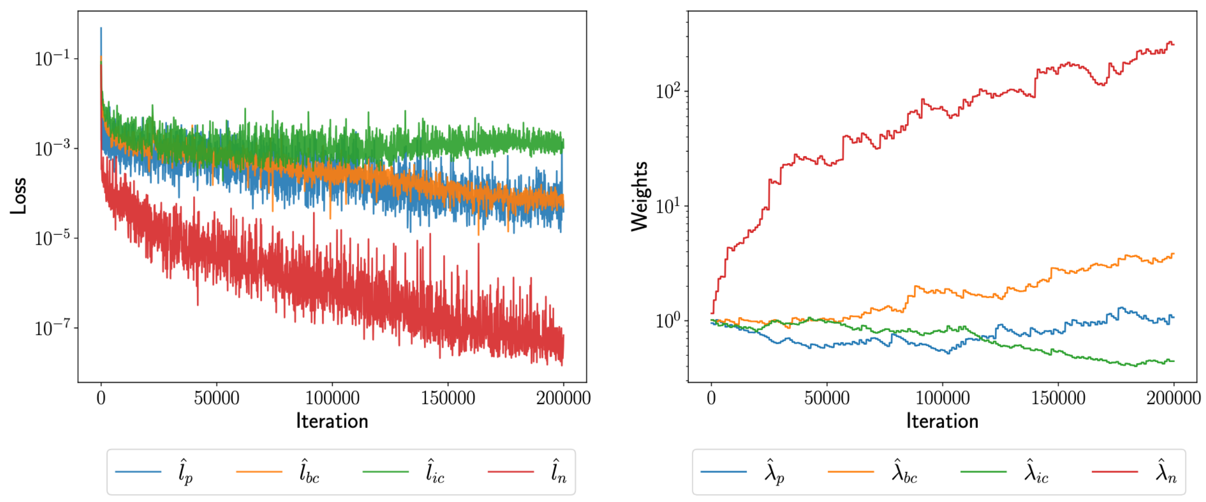
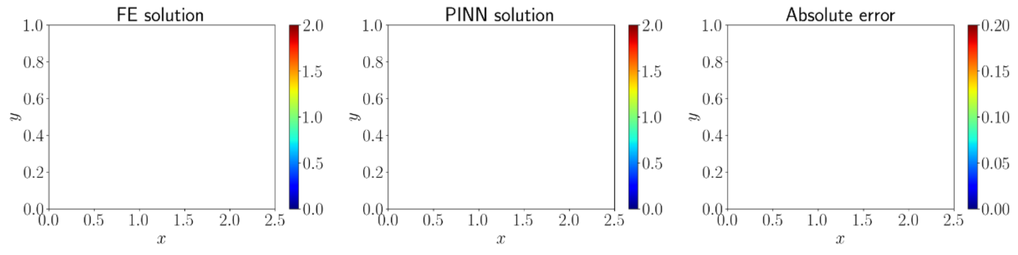
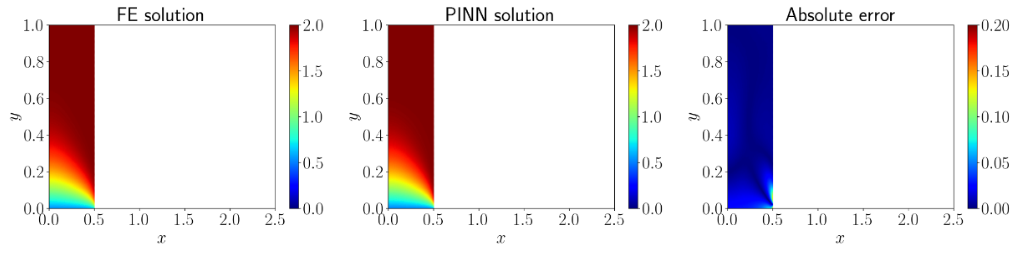


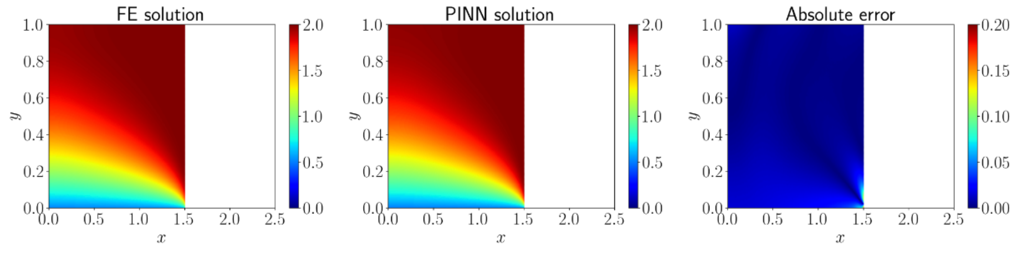
Figure 3:Loss terms and weight terms for example 1.



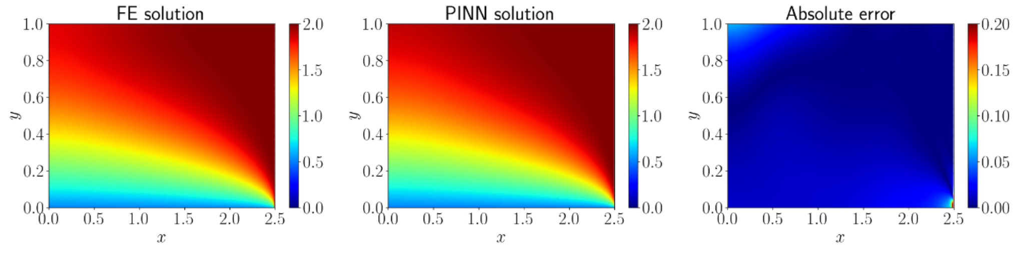
(a) At $t = 0.0$



(b) At $t = 0.5$



(c) At $t = 1.5$



(d) At $t = 2.5$

Figure 4: Comparison of FE solution to PINN solution for example 1.

Example 2: Geometry with evolving Dirichlet and Neumann boundary conditions

In this example we keep the same settings as the previous examples but we add a convection boundary condition given as

$$\frac{k_y \partial u}{\partial t} = -hu \; on \; \Gamma_N(t) \tag{24}$$

We choose the value of h=15 to promote significant heat losses from this boundary. The problem can be visualized in Fig. 5

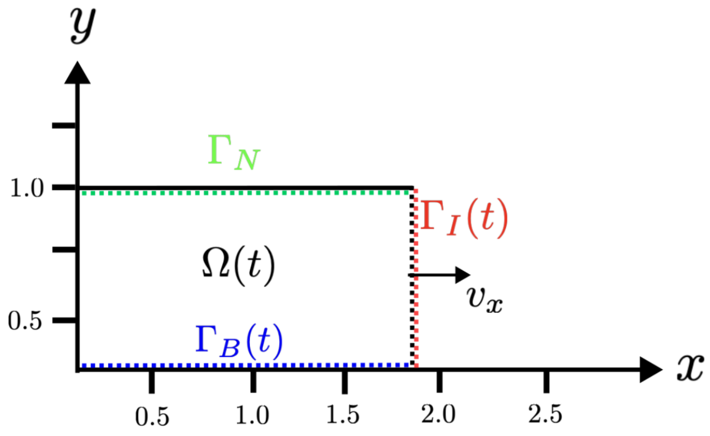


Figure 5: Illustration of example 2.

After non-dimensionalization, the loss is derived as

$$l = \lambda_p l_p + \lambda_{bc} l_{bc} + \lambda_{ic} l_{ic} + \lambda_{nc} l_{nc} \lambda_n l_n \tag{25}$$

$$\begin{aligned} = \frac{\lambda_p}{N_m N_p} \sum_{m=1}^{N_m} \sum_{i=1}^{N_P} \left[\tilde{\alpha}_x \frac{\partial^2 \tilde{u}_\theta}{\partial x^2} + \tilde{\alpha}_y \frac{\partial^2 \tilde{u}_\theta}{\partial \tilde{y}^2} - \frac{\partial \widetilde{u_\theta}}{\partial \tilde{t}} \right]^2 + \frac{\lambda_{bc}}{N_m N_{bc}} \sum_{m=1}^{N_m} \sum_{j=1}^{N_{bc}} \left[\tilde{u}_\theta - \frac{u_B}{u_I} \right]^2 \\ + \frac{\lambda_{ic}}{N_m N_{ic}} \sum_{m=1}^{N_m} \sum_{j=1}^{N_{ic}} \left[\tilde{u}_\theta - \frac{u_I}{u_I} \right]^2 + \frac{\lambda_{nc}}{N_m N_{nc}} \sum_{m=1}^{N_m} \sum_{j=1}^{N_{nc}} \left[\frac{\tilde{k}_y \tilde{u}_\theta}{\partial \tilde{t}} + h \tilde{u}_\theta \right]^2 + \frac{\lambda_n}{N_n} \sum_{n=1}^{N_n} \left[\tilde{\mathrm{k}}_{\mathrm{x}} \frac{\partial \tilde{u}_\theta}{\partial \tilde{x}} \right]^2 \end{aligned} \tag{26}$$

For this example, we use a DNN with 4 hidden layers and 128 neurons in each layer as the backbone model. We use the Tanh activation function. The parameters are initialized using the Glorot scheme, and the network parameters are trained using the ADAM optimizer with an initial learning rate of 5e-4, with an exponential decay at a rate of 0.9 every 1000 iterations. The total number of iterations is set to be 200000. The different batch sizes are chosen as $N_p = 8, N_p = N_b = N_{ic} = N_{nc} = N_n = 1024$.

The L^2 error is calculated only for the geometry that has been deposited. We summarize the average L^2 error over the entire time for the ablation studies in Table 2.

Table 2: Summary of ablation studies performed on example 2

Ablation Settings			Evaluation Metric	
Fourier Features	Modified DNN	Gradient Norm	Average Rel. L2 error	Computational Time (min)
✓	✓	✓	14.4×10^{-3}	45.2
✓	✓	✗	14.1×10^{-3}	44.2
✓	✗	✓	14.3×10^{-3}	28.3
✓	✗	✗	14.0×10^{-3}	27.6
✗	✓	✓	16.9×10^{-3}	36.8
✗	✓	✗	17.3×10^{-3}	35.8
✗	✗	✓	16.8×10^{-3}	23.6
✗	✗	✗	18.5×10^{-3}	23.4

Note that all cases have a low error, and we investigate the case with the lowest error. This is the case where the modified DNN is used with the Fourier features and no loss gradient balancing. The evolution of the loss and the weights for this case is shown in Fig.6 In Fig. 7, we plot the spatial temperature field at various time instances and observe that our PINN predictions are in close agreement with the FE solution.

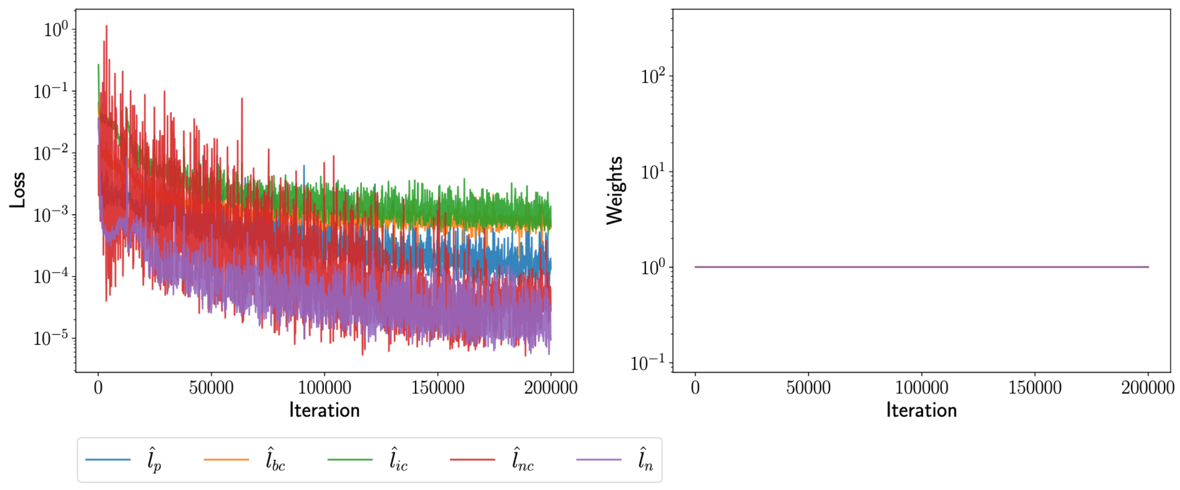
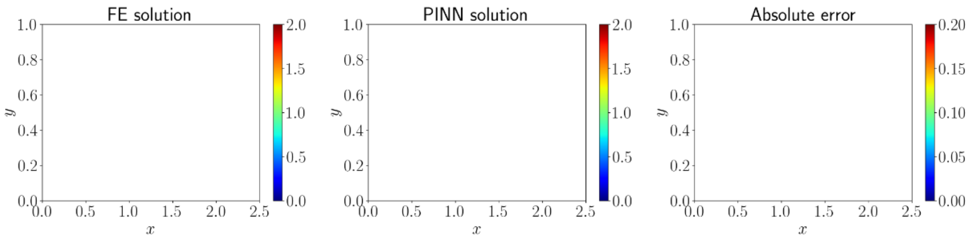


Figure 6: Loss terms and weight terms, for example, 2. Note that the weights are all one since this is the setting that attained the lowest error.

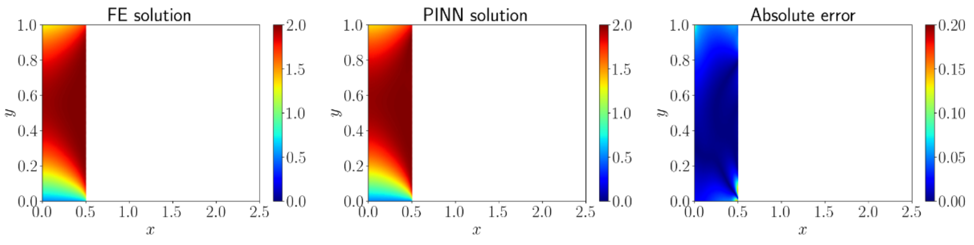
Conclusions

We presented a method to use PINNs in evolving geometries. We achieve this by proposing a new loss function that respects the causality of the addition. We performed systematic ablation studies to investigate the recent advancements made and their application to our work. Our method contrasts with the finite element formulation as it does not require spatial or temporal discretization. This underpins a major advantage of our method: we can query the solution at arbitrarily small time increments. Obtaining a similar temporal resolution using the finite element method can be computationally expensive. Further, our method is naturally biased to learn temporally causal solutions, facilitated by the decreases in collocation point density for increasing times. While our work presented a comprehensive analysis of the addition of a single bead, the question of depositing multiple beads still remains open. We are currently investigating methods to solve multiple bead addition without having to solve sequential problems. The sequential addition of layers was solved in the thesis work by [28]. It is essential to clarify that this method is not meant to replace classical solvers like the finite element method. For single, forward simulations, finite element solvers are more efficient and reliable. Further, we assumed that the geometry evolution is predefined. This is reasonable for extrusion-based additive manufacturing. However, for metal additive manufacturing, where the melt pool geometry depends on the laser power, modifications to our method are necessary [29]. Nonetheless, our work presents an opportunity to extend to parametric PDEs, which interests practicing engineers who require surrogate models for optimization, control, and uncertainty quantification. Therefore, our immediate next step is to evaluate the paradigm of physics-informed operator learning applied to evolving geometries.

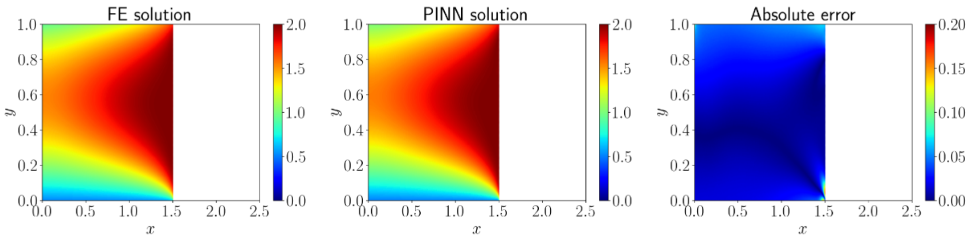
The true advantage of PINNs is when used to solve parametric or stochastic PDEs, as shown in [30]. However, the extension to parametric PDE can be achieved by integrating our proposed loss function over the parameter space and then finding an unbiased estimator for the same. We are currently extending this method to parametric PDEs and systematically evaluating its advantages and pitfalls.



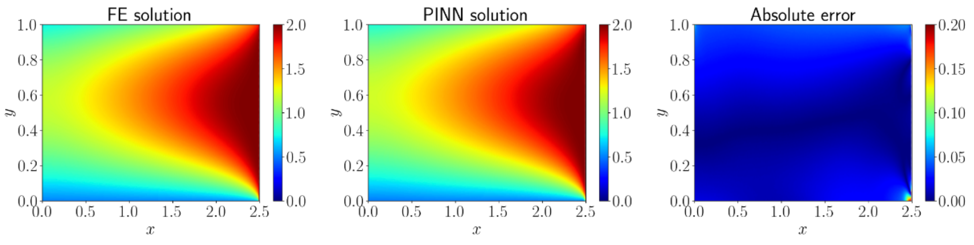
(a) At $t = 0.0$



(b) At $t = 0.5$



(c) At $t = 1.5$



(d) At $t = 2.5$

Figure 7: Comparison of FE solution to PINN solution for example 2.

Acknowledgments

Akshay J Thomas and Ilias Bilionis were supported by the AFOSR program on materials for extreme environments under grant number FA09950-22-1-0061. This research was also partly sponsored by the Composites Additive Manufacturing and Simulation (CAMS) consortium at Purdue University.

References

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

[2] D. C. Psichogios and L. H. Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE Journal*, vol. 38, no. 10, pp. 1499–1511, 1992.

[3] J. Bai, T. Rabczuk, A. Gupta, L. Alzubaidi, and Y. Gu, "A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics," *Comput Mech*, vol. 71, no. 3, pp. 543–562, 2023.

[4] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, "A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics," *Comput Methods Appl Mech Eng*, vol. 379, p. 113741, 2021.

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

[6] A. Henkes, H. Wessels, and R. Mahnken, "Physics informed neural networks for continuum micromechanics," *Comput Methods Appl Mech Eng*, vol. 393, p. 114790, 2022.

[7] L. Sun, H. Gao, S. Pan, and J.-X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," *Comput Methods Appl Mech Eng*, vol. 361, p. 112732, 2020.

[8] A. Mathews, M. Francisquez, J. W. Hughes, D. R. Hatch, B. Zhu, and B. N. Rogers, "Uncovering turbulent plasma dynamics via deep learning from partial observations," *Phys Rev E*, vol. 104, no. 2, p. 25205, 2021.

[9] Z. Fang and J. Zhan, "Deep physical informed neural networks for metamaterial design," *Ieee Access*, vol. 8, pp. 24506–24513, 2019.

[10] Z. Liu *et al.*, "Additive manufacturing of metals: Microstructure evolution and multistage control," *J Mater Sci Technol*, vol. 100, pp. 224–236, 2022.

[11] S. A. Niaki, E. Haghighat, T. Campbell, A. Poursartip, and R. Vaziri, "Physics-informed neural network for modeling the thermochemical curing process of composite-tool systems during manufacture," *Comput Methods Appl Mech Eng*, vol. 384, p. 113959, 2021.

[12] T. Würth, C. Krauß, C. Zimmerling, and L. Kärger, "Physics-informed neural networks for data-free surrogate modeling and engineering optimization–An example from composite manufacturing," *Mater Des*, vol. 231, p. 112034, 2023.

[13] S. Liao, T. Xue, J. Jeong, S. Webster, K. Ehmann, and J. Cao, "Hybrid thermal modeling of additive manufacturing processes using physics-informed neural networks for temperature prediction and parameter identification," *Comput Mech*, vol. 72, no. 3, pp. 499–512, 2023.

[14] J. Xie, Z. Chai, L. Xu, X. Ren, S. Liu, and X. Chen, "3D temperature field prediction in direct energy deposition of metals using physics informed neural network," *The International Journal of Advanced Manufacturing Technology*, vol. 119, no. 5, pp. 3449–3468, 2022.

[15] D. Kats, Z. Wang, Z. Gan, W. K. Liu, G. J. Wagner, and Y. Lian, "A physics-informed machine learning method for predicting grain structure characteristics in directed energy deposition," *Comput Mater Sci*, vol. 202, p. 110958, 2022.

[16] T. Yang, T.-N. Tsai, and J. Yeh, “A neural network-based prediction model for fine pitch stencil-printing quality in surface mount assembly,” *Eng Appl Artif Intell*, vol. 18, pp. 335–341, 2005, doi: 10.1016/j.engappai.2004.09.004.

[17] G. Tod, A. P. Ompusunggu, G. Struyf, G. Pipeleers, K. De Grave, and E. Hostens, “Physics-informed neural networks (PINNs) for improving a thermal model in stereolithography applications,” *Procedia CIRP*, vol. 104, pp. 1559–1564, 2021.

[18] E. Barocio, P. Pibulchinda, A. J. Thomas, V. Kapre, and A. Franc, “Validated Simulation for Large Scale Additive Manufacturing.,” *CAMX 2022*, 2022.

[19] P. Michaleris, “Modeling metal deposition in heat transfer analyses of additive manufacturing processes,” *Finite Elements in Analysis and Design*, vol. 86, pp. 51–60, 2014.

[20] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[21] S. Wang, S. Sankaran, and P. Perdikaris, “Respecting causality for training physics-informed neural networks,” *Comput Methods Appl Mech Eng*, vol. 421, p. 116813, 2024.

[22] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, “An expert’s guide to training physics-informed neural networks,” *arXiv preprint arXiv:2308.08468*, 2023.

[23] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[24] I. Goodfellow, *Deep Learning*. MIT Press, 2016.

[25] S. Wang, Y. Teng, and P. Perdikaris, “Understanding and mitigating gradient flow pathologies in physics-informed neural networks,” *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[26] M. Tancik *et al.*, “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains,” *Adv Neural Inf Process Syst*, vol. 2020-December, Jun. 2020, Accessed: Sep. 15, 2024. [Online]. Available: https://arxiv.org/abs/2006.10739v1

[27] C. L. Wight and J. Zhao, “Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks,” *arXiv preprint arXiv:2007.04542*, 2020.

[28] A. Jacob Thomas, “ACCELERATING COMPOSITE ADDITIVE MANUFACTURING SIMULATIONS: A STATISTICAL PERSPECTIVE,” Purdue University Graduate School, 2023.

[29] M. Ansari, A. Martinez-Marchese, Y. Huang, and E. Toyserkani, “A mathematical model of laser directed energy deposition for process mapping and geometry prediction of Ti-5553 single-tracks,” *Materialia (Oxf)*, vol. 12, p. 100710, 2020.

[30] S. Karumuri, R. Tripathy, I. Bilionis, and J. Panchal, “Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks,” *J Comput Phys*, vol. 404, p. 109120, 2020.