Shengze Cai[1]
Division of Applied Mathematics,
Brown University,
Providence, RI 02912
e-mail: shengze_cai@brown.edu

Zhicheng Wang[1]
Division of Applied Mathematics,
Brown University,
Providence, RI 02912
e-mail: zhicheng_wang@brown.edu

Sifan Wang
Graduate Group in Applied Mathematics and
Computational Science,
University of Pennsylvania,
Philadelphia, PA 19104
e-mail: sifanw@sas.upenn.edu

Paris Perdikaris[2]
Department of Mechanical Engineering and
Applied Mechanics,
University of Pennsylvania,
Philadelphia, PA 19104
e-mail: pgp@seas.upenn.edu

George Em Karniadakis[2]
Division of Applied Mathematics,
Brown University,
Providence, RI 02912
e-mail: george_karniadakis@brown.edu

Physics-Informed Neural Networks for Heat Transfer Problems

Physics-informed neural networks (PINNs) have gained popularity across different engineering fields due to their effectiveness in solving realistic problems with noisy data and often partially missing physics. In PINNs, automatic differentiation is leveraged to evaluate differential operators without discretization errors, and a multitask learning problem is defined in order to simultaneously fit observed data while respecting the underlying governing laws of physics. Here, we present applications of PINNs to various prototype heat transfer problems, targeting in particular realistic conditions not readily tackled with traditional computational methods. To this end, we first consider forced and mixed convection with unknown thermal boundary conditions on the heated surfaces and aim to obtain the temperature and velocity fields everywhere in the domain, including the boundaries, given some sparse temperature measurements. We also consider the prototype Stefan problem for two-phase flow, aiming to infer the moving interface, the velocity and temperature fields everywhere as well as the different conductivities of a solid and a liquid phase, given a few temperature measurements inside the domain. Finally, we present some realistic industrial applications related to power electronics to highlight the practicality of PINNs as well as the effective use of neural networks in solving general heat transfer problems of industrial complexity. Taken together, the results presented herein demonstrate that PINNs not only can solve ill-posed problems, which are beyond the reach of traditional computational methods, but they can also bridge the gap between computational and experimental heat transfer. [DOI: 10.1115/1.4050542]

1 Introduction

The application of machine learning (ML) techniques to heat transfer problems can be dated back to 1990s, when artificial neural networks (ANN) were used to learn the convective heat transfer coefficients [1] from data. In recent years, more advanced learning-based methods have been developed also aided by the improvement of the appropriate hardware, e.g., GPU technology, and have been beneficial to various heat transfer problems [2–4]. In particular, the convolutional neural network (CNN)—a widely used deep learning technique—has been successfully employed to make predictions from image-like data in complex and high-dimensional problems [5–8]. For example, a CNN proposed in Ref. [8] was used to predict the local heat flux of turbulent channel flows by feeding it with the wall-shear stress and the wall pressure. In other work, a CNN based on the U-net [9] architecture, employed a two-dimensional (2D) slice of time-averaged temperature field as input to predict a ridge pattern, which was used to quantify the fraction and time variability of turbulent heat transfer [5]. The aforementioned methods are mostly based on the supervised learning strategy, where a database with labels (i.e., ground-truth) is required to train the model. The advantage of such data-driven methods is that once trained, the prediction procedure will be very fast. However, the data generation is a big issue and the generalization (from one experimental condition to another) of the model is not guaranteed. In addition to the supervised learning, other approaches have been taken into account. For instance, an unsupervised learning strategy was applied for a conjugate thermal optimization problem [10]. More recently, deep reinforcement learning (RL) has also been applied to control thermal systems; for example, Ref. [11] showed that RL-based control is able to stabilize the conductive regime and bring the onset of convection up to a Rayleigh number $\mathrm{Ra} = 3 \times 10^4$, outperforming the state-of-the-art linear controllers. Also, Ref. [12] applied RL to several natural and forced convection problems, demonstrating that their RL algorithms can alleviate the heat transfer enhancement related to the onset of convection.

The aforementioned efforts do not directly take into account the underlying physics of heat transfer problems. To tackle this problem, a multitask learning approach is required, such as the framework of physics-informed neural networks (PINNs). This approach was first proposed for solving both forward and inverse problems described by a combination of some data and of partial differential equations (PDEs), and subsequently it was applied to various fluid mechanics problems [13–16] as well as heat transfer problems [16–20]. For instance, motivated by the limited understanding of the physical mechanisms responsible for the heat transfer enhancement in rough turbulent Rayleigh–Bénard convection, the authors in Ref. [19] applied the PINN framework to predict turbulent transport at $\mathrm{Ra} = 2 \times 10^7$ in a subdomain of a Rayleigh–Bénard cavity filled with water and bearing two square-based roughness elements placed on the hot plate. PINNs training benefited from a large direct numerical simulations database. The influence of the choice of the data acquisition and residuals sampling, in relation to the problem geometry and initial/boundary conditions, was also reported [19]. NVIDIA also has developed the code SimNet based on PINNs and applied it to solve multiphysics problems involving heat transfer of a FPGA heat sink inside a channel [16]; see also Sec. 5.2 below. Moreover, PINNs can be employed to quantify the flow fields of natural convection from

[1]Shengze Cai and Zhicheng Wang contributed equally to this work.
[2]Corresponding authors.

Contributed by the Heat Transfer Division of ASME for publication in the JOURNAL OF HEAT TRANSFER. Manuscript received February 8, 2021; final manuscript received March 3, 2021; published online April 21, 2021. Assoc. Editor: Portonovo S. Ayyaswamy.

data measurements, as demonstrated in Ref. [18]. In this article, the authors inferred the flow fields based on a number of temperature as well as velocity measurements, and investigated the influence of the data selection. The flexibility offered by PINNs has also been leveraged to tackle free boundary and Stefan problems; a general class of problems for which the evolution of unknown boundaries and moving interfaces has to be estimated concurrently with solving the underlying PDE. For example, Wang et al. [21] have demonstrated the effectiveness of PINNs in solving both forward and inverse problems Stefan problems involving multiphase interfaces; see also Sec. 4 below. More recently, the PINNs algorithm was applied to quantify the velocity and pressure fields of natural convection over an espresso cup from temperature data obtained by a background-oriented schlieren experiment [20], indicating that the method can be successfully used to deal with real experimental data. In addition to PINNs, an unsupervised learning approach with auto-encoder and image gradient was proposed in Ref. [22] to solve the heat equations on a chip. The principle of this method is similar to PINNs. However, by training the network with a set of data, the proposed framework can also generalize the trained network for predicting solutions for heat equations with unseen source terms. These algorithms, which aim to solve the heat equations, introduce the physical models to the neural networks, and train the model by minimizing a loss function involving the residuals of the governing equations. Therefore, the solutions can be determined with limited data or without any data except for the boundary and initial conditions.

In this article, we mainly review the applications of PINNs on inverse heat transfer problems in forced and mixed convection and on the two-phase Stefan problem with a moving interface. We also include two examples from industry on how to use PINNs in the thermal design of power electronics. The article is organized as follows. In Sec. 2, we provide an overview of PINNs, and in Sec. 3, we present two prototype problems of forced and mixed convection. In Sec. 4, we present the formulation of PINNs for the two-phase Stefan problem with some illustrative results, and in Sec. 5, we present two industrial applications of PINNs to power electronics by ANSYS and NVIDIA. We conclude in Sec. 6 with a brief summary and outlook.

2 Overview of Physics-Informed Neural Networks

In the context of PINNs, a neural network is used to approximate the solutions of PDEs, expressed as

$$\begin{aligned} \mathbf{u}_t + \mathcal{N}_{\mathbf{x}}[\mathbf{u}] &= 0, \quad \mathbf{x} \in \Omega,\ t \in [0,T] \\ \mathbf{u}(\mathbf{x},0) &= h(\mathbf{x}), \quad \mathbf{x} \in \Omega \\ \mathbf{u}(\mathbf{x},t) &= g(\mathbf{x},t), \quad \mathbf{x} \in \partial\Omega,\ t \in [0,T] \end{aligned} \tag{1}$$

where $\mathcal{N}_{\mathbf{x}}$ is a general linear or nonlinear differential operator; $\mathbf{x} \in \mathbb{R}^d$ and t are the spatial and temporal coordinates, respectively; Ω and $\partial\Omega$ denote the computational domain and the boundary; $\mathbf{u}(\mathbf{x},t)$ is the solution of the PDEs with initial condition $h(\mathbf{x})$ and boundary condition $g(\mathbf{x},t)$. We remark that this formulation can be easily generalized to higher-order PDEs since they can be written as systems of first-order PDEs.

Following the framework of PINNs proposed in Ref. [23], $\mathbf{u}(\mathbf{x},t)$ is approximated by a fully connected network, which takes the coordinates $(\mathbf{x},t)$ as inputs and outputs $\mathbf{u}_{NN}(\mathbf{x},t)$. The neural network is composed of multiple hidden layers, where the inputs of each hidden layer $(X = [x_1, x_2, \ldots, x_i])$ and outputs $(Y = [y_1, y_2, \ldots, y_j])$ are propagated through the network as

$$y_j = \sigma(w_{i,j}x_i + b_j) \tag{2}$$

where $w_{i,j}$ and b_j are trainable weights and biases, respectively; $\sigma(\cdot)$ is the activation function representing a simple nonlinear transformation. The parameters of the network can be trained by minimizing a composite loss function taking the form

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_b + \mathcal{L}_0 \tag{3}$$

where

$$\mathcal{L}_r = \frac{1}{N_r}\sum_{i=1}^{N_r}\left|\mathbf{u}_t(\mathbf{x}^i,t^i) + \mathcal{N}_{\mathbf{x}}[\mathbf{u}(\mathbf{x}^i,t^i)]\right|^2 \tag{4a}$$

$$\mathcal{L}_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|\mathbf{u}(\mathbf{x}^i,t^i) - g^i\right|^2 \tag{4b}$$

$$\mathcal{L}_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left|\mathbf{u}(\mathbf{x}^i,t^i) - h^i\right|^2 \tag{4c}$$

Here $\mathcal{L}_r$, $\mathcal{L}_b$, and $\mathcal{L}_0$ penalize the residuals of governing equations, the boundary conditions and the initial conditions, respectively; N_r, N_b, and N_0 are the numbers of data points for different terms. Notice that all loss terms are a function of the network weight and bias parameters, $w_{i,j}$ and b_j, respectively, the dependence on which has been omitted to favor notation simplicity. In addition, if some data are available inside the domain, an extra loss term indicating the mismatch between the predictions and the data can be taken into account

$$\mathcal{L}_{\text{data}} = \frac{1}{N_d}\sum_{i=1}^{N_d}\left|\mathbf{u}(\mathbf{x}^i,t^i) - \mathbf{u}^i_{\text{data}}\right|^2 \tag{5}$$

To compute the residuals for $\mathcal{L}_r$, derivatives of the outputs with respect to the inputs (i.e., $\mathbf{u}_t$ and $\mathcal{N}_{\mathbf{x}}[\mathbf{u}]$) are required. Such computation is achieved in the PINN framework using automatic differentiation in the deep learning code. Automatic differentiation relies on the fact that combining the derivatives of the constituent operations by the chain rule gives the derivative of the overall composition. This technique is a key enabler for the development of PINNs, and is the key element that differentiates PINNs from similar efforts in the early 1990s [24,25], which relied on manual derivation of back-propagation rules. Nowadays, automatic differentiation capabilities are well-implemented in most deep learning frameworks such as TensorFlow [26] and PyTorch [27], and it allows us to avoid tedious derivations or numerical discretization while computing derivatives of all orders in space–time. In Sec. 3, the problem-dependent governing equations, loss functions and the training configurations of PINNs are described case by case.

A schematic of the PINN framework is demonstrated in Fig. 1, in which a simple heat equation $u_t = \alpha u_{xx}$ is used as an example to show how to setup a PINN for heat transfer problems. As shown in Fig. 1(*a*), the fully connected neural network is used to approximate the solution $u(x,t)$, which is then applied to construct the residual loss $\mathcal{L}_r$, boundary conditions loss $\mathcal{L}_b$, and initial conditions loss $\mathcal{L}_0$. The parameters of the fully connected network are trained using gradient-descent methods based on the back-propagation of the loss function. We also demonstrate the data points used for different terms of the loss function, as shown in Fig. 1(*b*). Note that the residual points for computing the residual loss $\mathcal{L}_r$ can be randomly selected in the space–time domain and the numbers of points can be defined by the users.

3 Convection Heat Transfer: Unknown Thermal Boundary Conditions

In standard heat transfer handbooks and textbooks, one can find explicit expressions of the Nusselt number as a function of the Reynolds and Prandtl numbers, for several prototypical heat transfer problems, e.g., inside and around pipes, boundary layers, and other surfaces given that the thermal boundary condition is precisely specified, i.e., constant temperature or constant heat flux, e.g., see Refs. [28–30]. Moreover, at the present time, standard

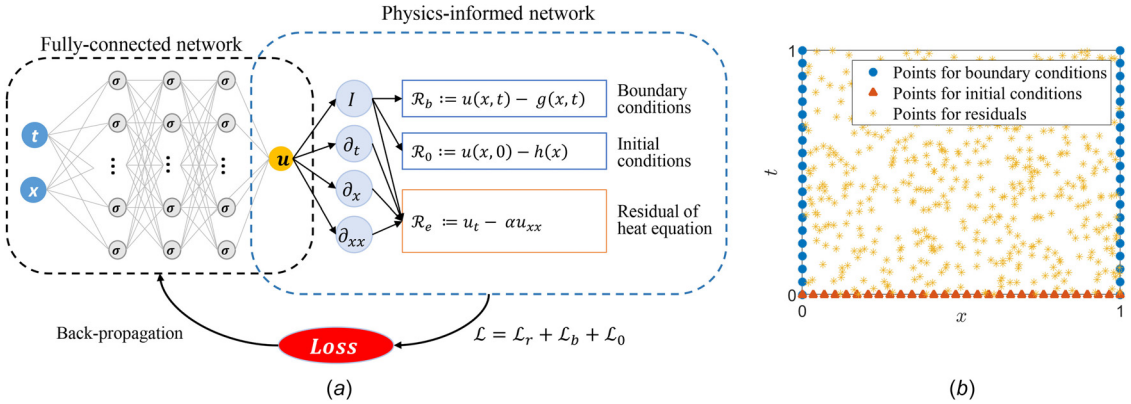


Fig. 1 Overview of physics-informed neural networks (PINNs). (*a*) Schematic of PINN framework. A fully connected neural network is used to approximate the solution $u(x, t)$, which is then applied to construct the residual loss $\mathcal{L}_r$, boundary conditions loss $\mathcal{L}_b$ and initial conditions loss $\mathcal{L}_0$. The derivatives of u are computed by automatic differentiation in TensorFlow [26]. The parameters of the fully connected network are trained using gradient-descent methods based on the back-propagation of the loss function. (*b*) Schematic of point selection for PINN in the computational domain. The points with different colors correspond to different terms in the loss function. Note that the residual points for computing the residual loss can be randomly selected in the space–time domain.

computational fluid mechanics (CFD) methods can simulate arbitrary thermal boundary conditions, e.g., the mixed Robin boundary condition, or arbitrary temperature distributions imposed on the heated surface with very good accuracy and on industrial complexity geometric domains. However, in real heat transfer applications, e.g., in power electronics or in a nuclear reactor, unlike the velocity boundary conditions, the thermal boundary conditions are never precisely known as this would require an enormous and complex instrumentation, which is not feasible in industrial applications but only in a few cases in research experimental setups.

The lack of thermal boundary conditions leads to an ill-posed boundary value problem for the energy equation, which cannot be solved no matter how sophisticated the CFD method is. To this end, we address this ill-posed problem here by asking the simple question: what if we have some temperature measurements at a few points, e.g., using simple thermocouples, in convenient locations that may or may not include the heated surface with the unknown boundary condition. Can we then solve an inverse problem using both the measurements and the governing equations of flow and heat transfer. In a typical CFD setup, this would require a tedious data assimilation method [31,32] blended in with flow and heat transfer solvers and may take extremely long times to converge, if it could converge at all. Here, inspired by the development of PINNs, we exploit the expressivity of deep neural networks to formulate this ill-posed problem, blending seamlessly data and mathematical models while simultaneously inferring both the flow and temperature fields.

3.1 Forced Convection. We first consider PINN applications to forced convection problems, which are commonly encountered in various industrial systems. The governing equations of this problem are the incompressible Navier–Stokes equations and the corresponding temperature equation

$$\begin{aligned} \frac{\partial \theta}{\partial t} + (\mathbf{u}\cdot\nabla)\theta &= \frac{1}{\mathrm{Pe}}\nabla^2\theta \\ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} &= -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{u} + \mathrm{Ri}\theta \\ \nabla\cdot\mathbf{u} &= 0 \end{aligned} \tag{6}$$

where θ, $\mathbf{u} = (u, v)^T$, and p are the dimensionless temperature, velocity, and pressure fields, respectively. Pe, Re, and Ri denote the Peclet, Reynolds, and Richardson numbers, respectively. Note that for forced convection discussed in this section, $\mathrm{Ri} = 0$. Given the boundary and initial conditions, simulating the forced convection problems with standard CFD techniques is relatively simple. However, here, we aim to demonstrate a new proof-of-concept of inferring the entire temperature field from very sparse temperature measurements. In particular, unlike the well-posed simulation problem, which explicitly defines the thermal boundary conditions on all boundaries of the domain, the problem we focus on is an ill-posed problem, where the boundary conditions are not fully known and have to be discovered from very few measurements while simultaneously infer the entire temperature field everywhere in the domain. Moreover, we can also infer the velocity and pressure fields everywhere in the domain. This can be considered as an inverse problem, which can be addressed by PINNs. To demonstrate the PINN effectiveness, we perform the forced convection simulations in two different scenarios: heat transfer inside an enclosure, and heat transfer over a stationary cylinder, where the flows are both steady.

3.1.1 Forced Convection in an Enclosure. We consider 2D steady forced convection heat transfer in an enclosure, as shown in Fig. 2; the two hot objects with red boundaries in the enclosure are at temperature $\theta = 1$; uniform cooling flow of $u = 1, v = 0$ enters from the left-bottom and exits from the right-top boundary. The temperature on the rest of the walls and inflow boundary is $\theta = 0$. On the outflow boundary, $\frac{\partial \theta}{\mathbf{n}} = 0$. The governing equations are given by Eq. (6) with $\mathrm{Re} = 50$, $\mathrm{Pe} = 36$, and $\mathrm{Ri} = 0.0$. Note that here the height of AB is used as the characteristic length for the Reynolds number. The simulation started from initial

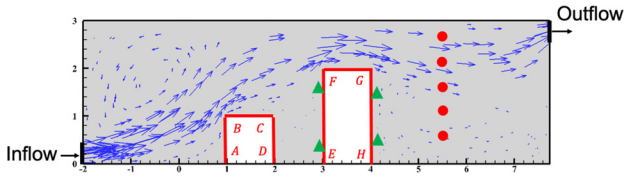


Fig. 2 Forced convection in an enclosure. The boundaries in red color are at higher temperature, and the cooling inflow enters from the left-bottom and exits at the right-top. The blue arrows show the velocity vectors. The temperature at the boundaries EF and GH are not known. The green triangles represent the temperature probes while the red circles denote the velocity probes.

conditions: $u = 0$, $v = 0$, and $\theta = 0$ and stopped when both the flow and temperature reach steady-state.

3.1.2 Flow Past Cylinder. Here, we consider the classical two-dimensional heat transfer problem of forced heat convection around a circular cylinder in steady-state; we assume that we have some data and the reference solution is obtained numerically using the spectral/*hp* element method [33]. The simulation domain size is $[-7.5D, 22.5D] \times [-10D, 10D]$, consisting of 2094 quadrilateral elements, where D is the diameter of the cylinder. The cylinder center is located at (0,0). On the fluid flow part, the cylinder surface is assumed to be no-slip, no-penetration wall. Uniform velocity $(u = U_\infty, v = 0)$ is imposed on the inflow boundary where $x/D = -7.5$, periodic boundary condition is used on the lateral boundaries where $y/D = \pm 10$, and zero-pressure boundary is prescribed on the outflow boundary where $x/D = 22.5$. On the heat transfer part, constant temperature $\theta = \theta_\infty = 0$ is imposed on the inflow boundary, periodic boundary is assumed on the lateral boundaries and zero-gradient is used on the outflow boundary. For the cylinder surface, constant wall temperature is considered as a simple example but any arbitrary temperature distribution can be readily modeled too. The governing equations of this problem are given in Eq. (6), with the Reynolds number $\mathrm{Re} = 20$ and Peclet number $\mathrm{Pe} = 200$. The simulation has been carried out until both flow and temperature reach steady-state.

3.1.3 Active Sensor Placement. In data-driven problems, it is often the case that the sensor locations, where data are collected, may affect the inferred results. To obtain the best sensor locations, it generally requires trial-and-error, which is a costly process. Therefore, we also propose a method to adaptively select the location of sensors to minimize the number of temperature measurements. This method allows us to start with a small number of fixed sensors (e.g., 4 or 5) and iteratively add more sensors at the essential locations.

The criterion of adding sensors we propose is based on the residual of the temperature equation in Eq. (6), i.e.,

$$e = u\theta_x + v\theta_y - \mathrm{Pe}^{-1}(\theta_{xx} + \theta_{yy}) \tag{7}$$

The reasons of using this residual include: (a) this residual, which represents the error of the temperature equation, can be directly computed by the neural network and it does not require any prior knowledge; (b) as we will show in the following result, there is a correlation between this residual and the posterior temperature error; and (c) this can be considered as a sensitivity metric of the temperature with respect to the location (x, y). Taking the forced convection of flow past cylinder as an example, the proposed method for adaptive sensor placement can be summarized as follows:

Step 1 Provide an initial sensor configuration.
Step 2 Train the neural network (PINN).
Step 3 Compute the residual (7) on the cylinder boundary inferred by PINN and find the position with maximum value of Eq. (7).
Step 4 Update the sensor placement and training data, and go back to step 2.

3.1.4 Results. For the steady forced convection problem, the PINN loss function is expressed as

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_{ub} + \mathcal{L}_{\theta b} + \mathcal{L}_\theta \tag{8}$$

where

$$\mathcal{L}_r = \frac{1}{N_r}\sum_{k=1}^{4}\sum_{i=1}^{N_r} |e_k(x^i, y^i)|^2 \tag{9a}$$

$$\mathcal{L}_{ub} = \frac{1}{N_{ub}}\sum_{i=1}^{N_{ub}} \left[\mathbf{u}(x^i, y^i) - \mathbf{u}_b^i\right]^2 \tag{9b}$$

$$\mathcal{L}_{\theta b} = \frac{1}{N_{\theta b}}\sum_{i=1}^{N_{\theta b}} |\theta(x^i, y^i) - \theta_b|^2 \tag{9c}$$

$$\mathcal{L}_\theta = \frac{1}{N_\theta}\sum_{i=1}^{N_\theta} |\theta(x^i, y^i) - \theta_{\mathrm{data}}^i|^2 \tag{9d}$$

The first term $\mathcal{L}_r$ penalizes the governing equations (6), including the heat equation, the momentum equations, and the continuity equation. Here, N_r is the batch size of residual points, which are randomly selected in the spatial domain. The second and third terms are the boundary conditions for velocity and temperature fields, respectively. It should be noted that the residual points for velocity boundary and temperature boundary can be different, thus two independent terms are applied in the loss function. θ_b denotes the environmental temperature, which is known in practice. The last term of the loss function $\mathcal{L}_\theta$ is the mismatch between the inferred temperatures and the in situ measured values, e.g., at the thermocouple locations. The value of N_θ depends on the sensor placement in different scenarios.

The parameters of the neural networks are randomly initialized using the Glorot scheme [34] and trained by the Adam optimizer [35] with a decreasing learning rate schedule. The results are obtained after 80,000 iterations with decreasing learning rates of 1×10^{-3}, 1×10^{-4}, 1×10^{-5}, and 1×10^{-6} (20k iterations for each).

To investigate the performance of the proposed neural network on inferring the temperature, velocity, and pressure fields in the domain, the results obtained by PINNs are quantitatively evaluated against the reference solutions (provided by the high-order CFD method). We use the relative L_2 errors as the metric

$$\varepsilon_V = \| V - V^* \|_2 / \| V^* \|_2 \tag{10}$$

where V represents one of the predicted quantities (θ, u, v, p) and V^* is the corresponding reference.

Enclosure. We employ a fully connected neural network with ten hidden layers and 120 neurons per layer. The number of residual points for the three terms in loss function 8 are $N_r = 10{,}736$, $N_{ub} = 757$, and $N_{\theta b} = 548$. Note that the points for $\mathcal{L}_{ub}$ include all the boundaries of the enclosure since the velocity boundary conditions are always well defined. As shown in Fig. 2, the points of $N_{\theta b}$ are uniformly distributed in all other boundaries except at EF and GH. Moreover, to mimic the scenario in real life that limited measurement data are available, here we assume that the temperature at the positions of the four green triangles and velocity at the positions of the five circles are known, see Fig. 2. The comparison between PINN-inferred result and the reference simulation result is shown in Fig. 3. It could be observed that PINN can predict the flow and temperature in the whole domain well qualitatively. The L_2 error of the temperature on the two unknown walls EF and GH is 10.82%.

Flow past cylinder. A fully connected neural network with ten hidden layers and 200 neurons per layer is employed in PINNs to infer the solutions. The numbers of residual points for different terms in loss function (8) are given as $N_r = 20{,}000$, $N_{ub} = 900$, and $N_{\theta b} = 500$. In particular, the points for computing $\mathcal{L}_{ub}$ include the inlet, upper, and lower bounds, as well as the cylinder boundary. We should point out again that the boundary condition of circular cylinder for velocity is known (no-slip) and that for temperature is unknown. The number of measurements (N_θ) depends on case by case and is specified in the following. In the following, we consider first the case of (unknown) prescribed temperature and subsequently the case of prescribed heat flux on the cylinder surface.

Constant temperature surface. Nine different sensor configurations are considered for the case with constant temperature surface, which are illustrated in Fig. 4. The number of temperature measurements N_θ varies from 5 to 11. As shown in the figure, we only have a couple of sensors on the cylinder and a couple more

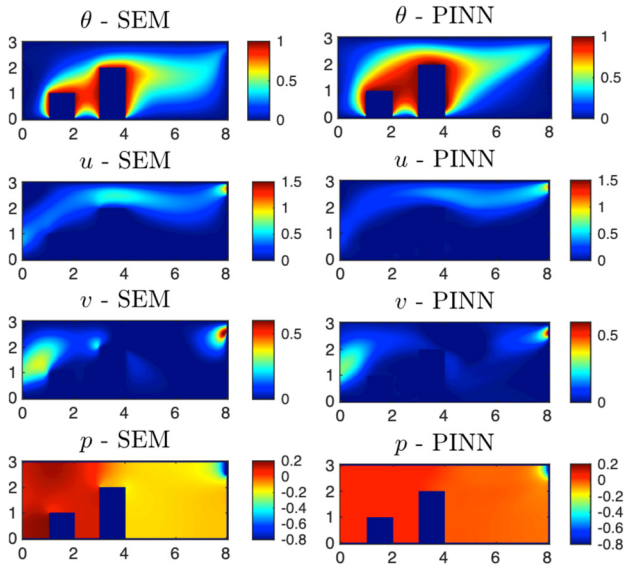


Fig. 3 Forced convection in an enclosure: comparison of θ, u, v, and p inferred by PINN and the reference simulation results by the spectral element method

in the wake. Based on these observations, the thermal boundary condition on the cylinder surface is to be inferred. The relative L_2 errors of temperature, velocity, and pressure fields computed over the whole domain are also provided in Fig. 4, along with the sensor placements. We note that the training of the network parameters is a nonconvex optimization problem (where a global minima is not guaranteed), thus that the result of each PINN simulation may be affected by the randomness of network initialization. Therefore, we perform ten independent training processes for each configuration and demonstrate the best results in Fig. 4.

For the first case shown in the figure, there are only five sensors: two on the cylinder surface and three in the wake. Although the neural network can predict accurate solutions of velocity and pressure, it fails to predict the temperature field very accurately ($\varepsilon_\theta > 10\%$). As seen from case 2, adding one more sensor on the surface can dramatically improve the results, showing that the information at the front stagnation point of the cylinder is significant in this problem. Case 3 is used to investigate another sensor placement in the wake behind the cylinder, which differs from case 2 and we can find that it can further increase the accuracy. The second group in Fig. 4, including cases 4, 5, and 6, show the fact that we can obtain relatively good results when there is no temperature measurement on the cylinder surface. Increasing the number of sensors around the cylinder can improve the inference performance. Moreover, compared with the third group (cases 7, 8, and 9), we find that placing one more sensor at the front stagnation point on the cylinder surface can help the method to find a much more accurate solution. For example, the temperature error in case 7 is only 1.25% while that in case 4 is 15.09%. It is worth noting that in all different sensor configurations, the velocity and pressure fields are accurately inferred by the PINN algorithm. The reason is that the boundary conditions of the velocity are well-defined, thus the Navier–Stokes (NS) equations (which are independent to the heat equation) can be well-resolved by PINNs [15].

A typical example of the inferred results is demonstrated in Fig. 5, where the fields are inferred by PINN with sensor configuration case 3. The pointwise absolute errors between CFD simulation and PINN results are also given, where we can see the magnitudes of the errors are very small compared to the magnitudes of inferring quantities. Figure 6 demonstrates the corresponding temperature and Nusselt number profiles on the cylinder surface. We observe that the temperature profile is approximately constant (equal to 1) and the Nusselt number looks consistent with the CFD solution. The mean Nusselt numbers along the cylinder surface of exact and predicted solutions are 5.90 and 5.89, respectively. That results in an accurate Nusselt number prediction with error smaller than 1%. Note that there are only six sensors used in

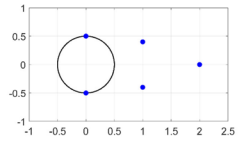
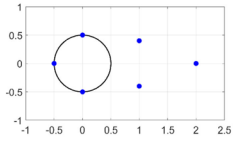
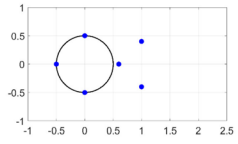
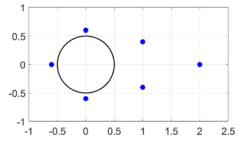
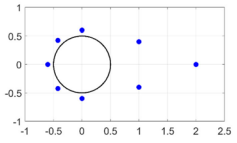
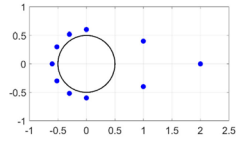
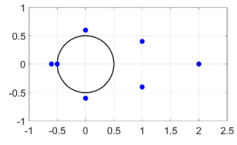
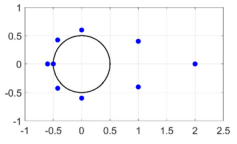
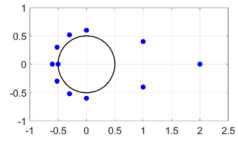
Case 1		Case 2		Case 3	
Sensor placement	Errors	Sensor placement	Errors	Sensor placement	Errors
	ϵ_u: 0.30 % ϵ_v: 0.39 % ϵ_p: 0.72 % ϵ_θ: 11.61 %		ϵ_u: 0.50 % ϵ_v: 0.70 % ϵ_p: 1.17 % ϵ_θ: 2.43 %		ϵ_u: 0.30 % ϵ_v: 0.47 % ϵ_p: 0.82 % ϵ_θ: 0.73 %
Case 4		**Case 5**		**Case 6**	
Sensor placement	Errors	Sensor placement	Errors	Sensor placement	Errors
	ϵ_u: 0.51 % ϵ_v: 0.95 % ϵ_p: 1.40 % ϵ_θ: 15.09 %		ϵ_u: 0.30 % ϵ_v: 0.49 % ϵ_p: 0.84 % ϵ_θ: 5.64 %		ϵ_u: 0.29 % ϵ_v: 0.47 % ϵ_p: 0.84 % ϵ_θ: 4.85 %
Case 7		**Case 8**		**Case 9**	
Sensor placement	Errors	Sensor placement	Errors	Sensor placement	Errors
	ϵ_u: 0.22 % ϵ_v: 0.32 % ϵ_p: 0.59 % ϵ_θ: 1.25 %		ϵ_u: 0.33 % ϵ_v: 0.52 % ϵ_p: 0.88 % ϵ_θ: 1.39 %		ϵ_u: 0.27 % ϵ_v: 0.40 % ϵ_p: 0.74 % ϵ_θ: 1.45 %

Fig. 4 Forced convection of flow past cylinder: results of different sensor configurations for inferring constant temperature surface

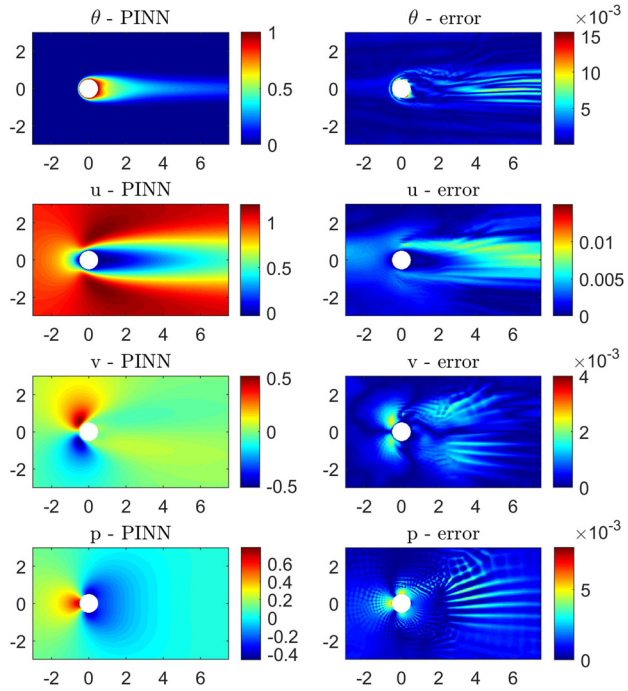


Fig. 5 Forced convection of flow past a cylinder with constant temperature surface: temperature, velocity and pressure fields inferred by PINN with sensor configuration Case 3 (left) and the pointwise absolute errors between CFD and PINN (right)

case 3. Using such limited measurements for prediction is an inverse problem, which is generally difficult to solve. In addition to the unknown boundary condition, the proposed neural network can simultaneously infer the velocity, pressure, and temperature fields everywhere in the domain.

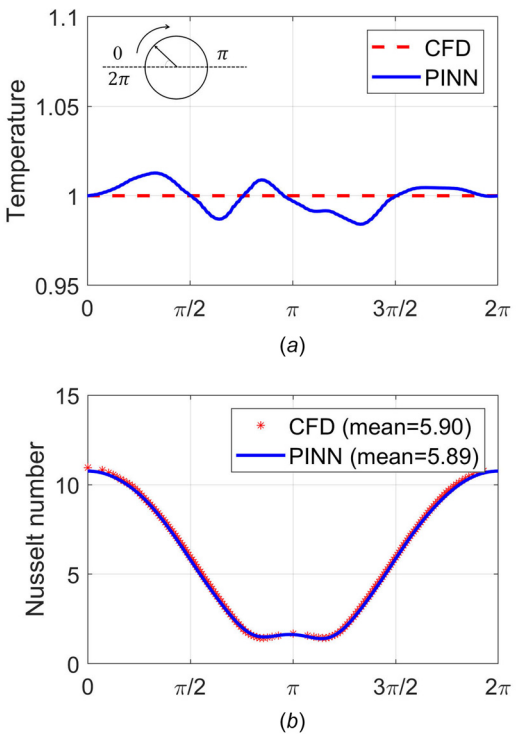


Fig. 6 Forced convection of flow past a cylinder with constant temperature surface: (*a*) temperature and (*b*) Nusselt number profiles on the cylinder boundary, inferred by PINN with sensor configuration Case 3

Constant flux surface. To discover the boundary condition with constant heat flux ($d\theta/d\mathbf{n} = 0$), we apply an experimental setup with seven sensors (three on the boundary and four in the wake domain), which is illustrated at the top-left corner in Fig. 7. The corresponding temperature and Nusselt number profiles are also demonstrated in the figure, where we can observe the consistency between CFD simulation and PINN inference. The mean Nusselt numbers along the cylinder surface of CFD and PINN solutions are 7.11 and 6.96, respectively. In this case, the L_2 errors of (θ, u, v, p) inferred by PINN are $3.18\%, 0.18\%, 0.24\%$, and 0.46%, respectively.

Active sensor placement. We have introduced the method for active sensor placement in a previous section, and here we present an example of inferring the constant temperature surface in the flow past a cylinder to demonstrate the effectiveness of this method. As mentioned, the residual of heat transfer equation is considered as the criterion since there exists a correlation between the residual and the error of Nusselt number (which is a posterior assessment). As an example for explanation, the normalized error of Nusselt number and the normalized residual of the network for case 1 in Fig. 4 are illustrated in Fig. 8. As we can see, although the shapes of the curves are not exactly identical to each other, the peak locations of the residual are close to the locations with maximum Nusselt number error. Therefore, it is reasonable to expect better performance of the PINN if we add new sensors on the position with maximal residual.

Based on this strategy, we carry out an experiment of active sensor placement for forced convection of flow past a cylinder. In this case, the initial sensor setup is case 1 in Fig. 4, which contains five measurements. As shown in Fig. 9(*a*), the algorithm automatically added a sensor near the front stagnation point, which results in the configuration corresponding to case 2 in Fig. 4. That means the active sensor placement algorithm with PINNs can achieve a very high accuracy only with one iteration in this case. From Figs. 9(*b*) and 9(*c*), we can observe that the profiles of temperature

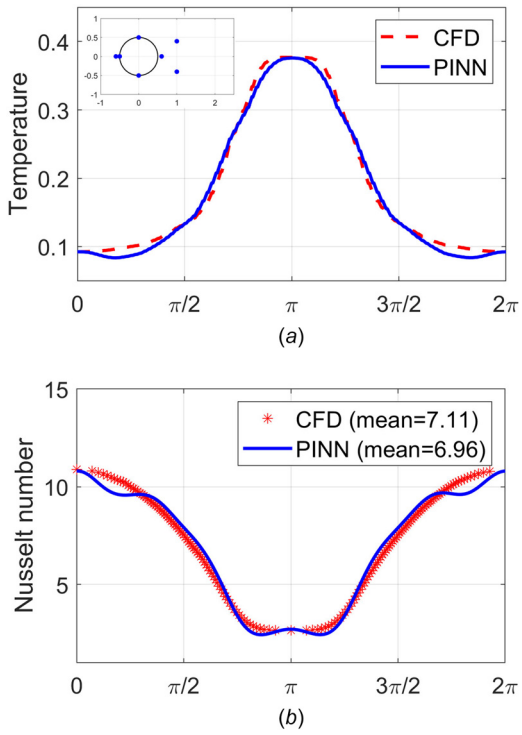


Fig. 7 Forced convection of flow past a cylinder with constant flux surface: (*a*) temperature and (*b*) Nusselt number profiles on the cylinder boundary. The sensor configuration is given at the top-left corner. With this setup, the errors of (θ, u, v, p) inferred by PINN are 3.18%, 0.18%, 0.24%, and 0.46%, respectively.

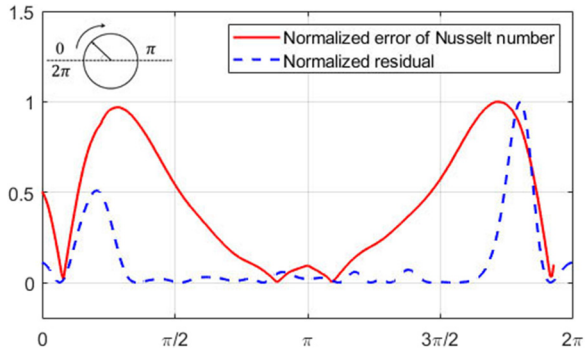


Fig. 8 Active sensor placement: the normalized error of Nusselt number and the normalized residual of the heat equation on the cylinder surface, inferred by PINN with sensor configuration Case 1 in Fig. 4. We can observe the correlation between the two profiles.

and Nusselt number on the cylinder surface are consistent with the CFD solutions. The L_2-norm error of temperature in the domain dramatically decreases to about 3% after the first iteration. We note that although the active sensor placement algorithm can be performed with more iterations, here we only demonstrate one iteration since the accuracy is already high enough to terminate the process. For complex problems, more iterations (i.e., more sensors) are expected. However, such adaptive sensor placement algorithm allows us to avoid a trial-and-error procedure for selecting sensor locations.

3.2 Mixed Convection. To obtain data and the reference solution, we consider a two-dimensional heat transfer problem of the mixed convection around a circular cylinder and we numerically simulate it using the spectral/*hp* element method, with the computational domain identical to the one described in Sec. 3.1.2. The cylinder surface is assumed to be no-slip, no-penetration wall, and constant wall temperature is considered. The governing equations of this problem are the Boussinesq approximations of the incompressible Navier–Stokes equations and the corresponding heat transfer equation, as shown in Eq. (6), where the Reynolds number $\mathrm{Re} = 100$, Péclet number $\mathrm{Pe} = 71$, and Richardson number $\mathrm{Ri} = 1.0$. In this article, we investigate the mixed convection where the force term is acted in the $-y$ direction. Compared to the forced convection investigated in the previous section, the mixed convection problem in this section involves unsteady flow. To assess the PINN method, the time span of interest is about $t \in [0, 15]$, covering more than three shedding periods, and the time-step is $\Delta t = 0.1$. Moreover, there is a force term in the NS equations, which couples the solutions of temperature and velocity fields. The mixed convection problem we consider is still an ill-posed problem, where the temperature boundary condition on the cylinder surface and the entire flow fields need to be inferred from a few measurements only. In addition to the thermocouples, in this section, we assume that it is available to measure a small patch of temperature in the wake behind the cylinder, e.g., using a technique such as digital particle image thermometry (DPIT) [36].

Results. For the unsteady case, PINN takes the space and time coordinates (t, x, y) as inputs and outputs (θ, u, v, p). Different from the network used for forced convection problem, here we employ the formulation of neuronwise locally adaptive activation function [37], where the relation between the input X and outputs Y of the lth each hidden layer is expressed as

$$y_j = \sigma(\alpha_j^l(w_{i,j}^l x_i^l + b_j^l)) \tag{11}$$

where α_j is a neuron-level parameter that can be learned during the training process and $\sigma(\cdot) = \sin(\cdot)$. The unknown parameters of the neural network model can be learned by minimizing the following loss function

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_{ub} + \mathcal{L}_{\theta b} + \mathcal{L}_\theta + \mathcal{L}_{\mathrm{sym}} \tag{12}$$

where

$$\mathcal{L}_r = \frac{1}{N_r}\sum_{k=1}^{4}\sum_{i=1}^{N_r} |e_k(x^i, y^i, t^i)|^2 \tag{13a}$$

$$\mathcal{L}_{ub} = \frac{1}{N_t N_{ub}}\sum_{i=1}^{N_t N_{ub}} |\mathbf{u}(x^i, y^i, t^i) - \mathbf{u}_b^i|^2 \tag{13b}$$

$$\mathcal{L}_{\theta b} = \frac{1}{N_t N_{\theta b}}\sum_{i=1}^{N_t N_{\theta b}} |\theta(x^i, y^i, t^i) - \theta_b|^2 \tag{13c}$$

$$\mathcal{L}_\theta = \frac{1}{N_t N_\theta}\sum_{i=1}^{N_t N_\theta} |\theta(x^i, y^i, t^i) - \theta_{\mathrm{data}}^i|^2 \tag{13d}$$

$$\mathcal{L}_{\mathrm{sym}} = \frac{1}{N_t N_{\mathrm{sym}}}\sum_{i=1}^{N_t N_{\mathrm{sym}}} |\theta(x^i, y^i, t^i) - \theta(x^i, -y^i, t^i)|^2 \tag{13e}$$

Here N_r, $N_t N_{\theta b}$, $N_t N_{ub}$, $N_t N_\theta$, and $N_t N_{\mathrm{sym}}$ represent the number of residual points corresponding to different terms, with N_t the number of snapshots. The first term enforces the governing equations by minimizing the residuals of Eq. (6). Here, N_r training points are randomly selected in the spatiotemporal domain, in the case we have the batch size $N_r = 10{,}000$ for each iteration. The second and third terms are the boundary conditions for velocity and temperature fields, respectively. Note again that the temperature surface of the cylinder boundary is unknown, while other boundaries

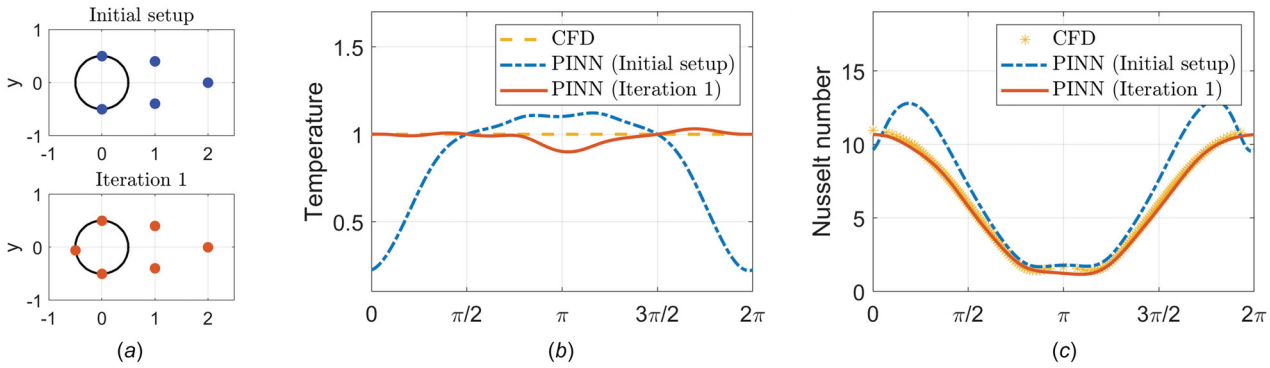


Fig. 9 Active sensor placement: (*a*) the sensor configurations of initial setup and after the first iteration, (*b*) temperature profile on the cylinder surface, and (*c*) Nusselt number on the cylinder surface

including the inlet, the upper and lower bounds of the computational domain are given as θ_b.

We aim to infer the entire flow fields from temperature measurements at a few locations (e.g., thermocouples) as well as in a small area (e.g., DPIT patch). An example of the configuration is demonstrated in Fig. 10(*a*), where DPIT measures the temperature in $[1,2]\times[-0.5,0.5]$, which is represented by a 8×8 grid. Together with three additional sensors, we have $N_\theta = 67$ for the PINN algorithm to infer the entire fields. Compared to the loss function used for forced convection problems, here we also add an extra term $\mathcal{L}_{\text{sym}}$, which is used to enforce the symmetric nature of the temperature profile on the cylinder surface. Although the loss function without $\mathcal{L}_{\text{sym}}$ also works, we find that this a priori knowledge can help to improve the accuracy. Here, we also note that it is flexible to encode any prior knowledge in the PINN framework. We randomly select $N_{\text{sym}} = 50$ on the cylinder surface to enforce the symmetry condition.

The neural network applied for mixed convection problem is composed of ten hidden layers and 150 neurons per layer. The training procedure is performed by applying the Adam optimizer with decreasing learning rates 1×10^{-3}, 1×10^{-4}, 1×10^{-5}, and 5×10^{-6}; each involves 200k iterations. This can ensure that the neural network converges to a plateau after training.

The relative L_2-norm errors of inference results over the spatial domain are demonstrated in Fig. 10(*b*), which indicates that the errors are almost constant in time. We observe that the temperature error for the entire domain is approximately 2%, while the maximum error on the cylinder surface is roughly 5%, as shown in Fig. 10(*c*). The 2D flow fields at $t = 5.0$ are also illustrated in Fig. 10(*d*), along with the pointwise absolute errors. Overall, we can see the magnitudes of the errors are very small. However, it is also worth noting that the errors of the flow (velocity and pressure) mainly exist in the region far from the cylinder, while the mismatch of the temperature is near the cylinder. The reason is that the velocity boundary condition on the cylinder surface is known (no-slip) while that for temperature is unknown. We also expect that adding velocity sensors in the downstream region will further improve the inference performance of PINNs, and this is a straightforward extension for the interested readers to pursue.

4 Two-Phase Stefan Problems

A large class of problems in heat transfer involve dynamic interactions between different material phases, giving rise to moving boundaries or free interfaces. A classical example is the so-called Stefan problem that traces back to the ice solidification problem describing the joint evolution of a liquid and a solid phase related to heat transfer [38]. This problem was also considered in 1831 by Lamé and Clapeyron in relation to the problems of ice formation in the polar seas [39]. Applications of free boundary and Stefan problems are ubiquitous in science and engineering as they can naturally model continuum systems with phase transitions, moving obstacles, multiphase dynamics, competition for resources, etc. Specific use cases in the context of heat transfer include thermal convection [40], Marangoni convection in a liquid pool [41], chemical vapor deposition [42], crystal growth and solidification [43–46], welding [47,48], semiconductor design [49], and beyond [42].

Since their initial conception nearly two centuries ago, free boundary and Stefan problems now define a well-studied area in applied mathematics both in terms of theory [50], numerical methods [51], and applications across a wide range of problems in science and engineering [42]. A unique characteristic of free boundary problems that introduces great challenges in terms of computational modeling is that they necessitate the solution of PDEs in domains with unknown boundaries and complex time-dependent interfaces. Different numerical methods have been developed to solve various types of free boundary problems, giving rise to different approaches for resolving the evolution of moving boundaries and free dynamic interfaces [52–57]. All these methods have their own advantages and limitations, and have been proven effective for certain classes of Stefan problems. However, these classical techniques are usually specialized to a specific type of free boundary problem and cannot be easily

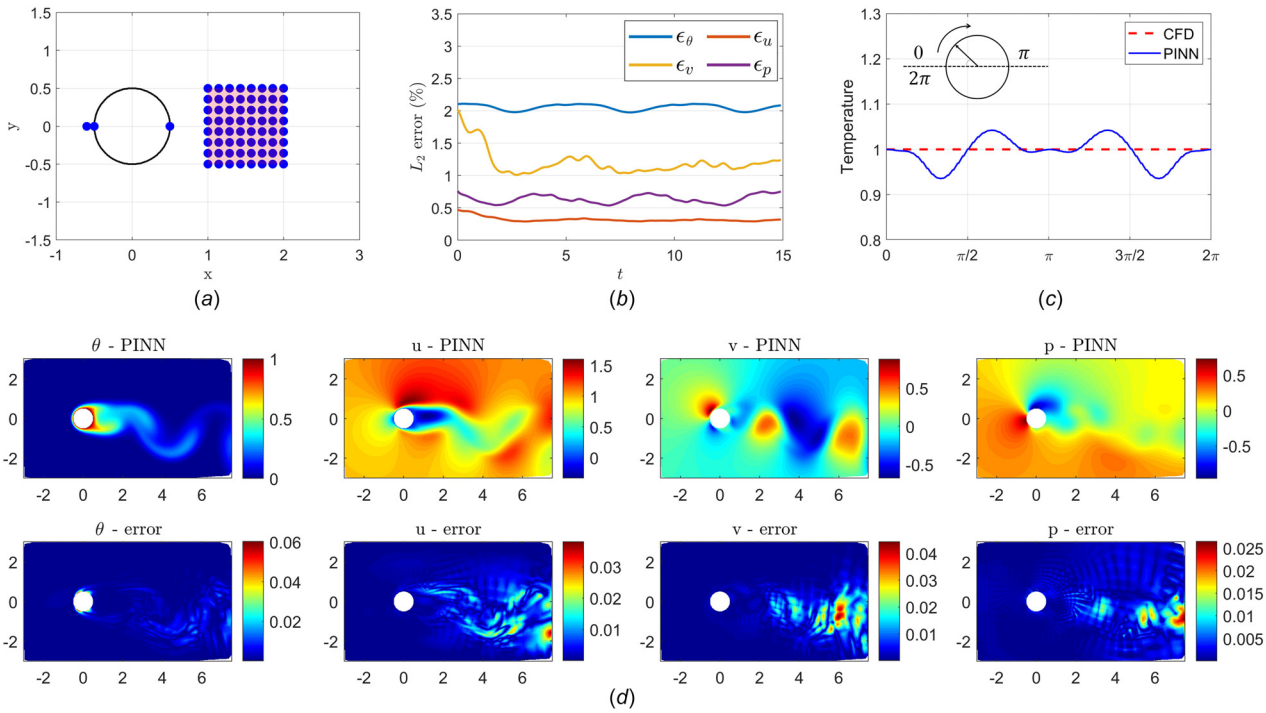


Fig. 10 Mixed convection of flow past a cylinder. (*a*) Temperature measurements. DPIT measures temperature in an area represented by a red rectangle with 8×8 grids. (*b*) The relative L_2-norm errors of temperature, velocity, and pressure fields along time inferred by PINN. (*c*) The mean temperature profile on the cylinder surface. (*d*) The 2D fields at $t = 5.0$ inferred by PINN and the corresponding pointwise error maps.

adapted to build a general framework for seamlessly synthesizing governing physical laws and observational data.

In this section, we will illustrate how PINN models can be adapted to handle free boundaries and time-dependent interfaces using a pedagogical example in heat conduction to illustrate the main ideas. To this end, let us consider heat transfer in a two-phase system, where the latent temperature distributions within each of the two phases, $u_1(x,t)$ and $u_2(x,t)$, respectively, satisfies a heat equation

$$\frac{\partial u_i}{\partial t} = k_i \frac{\partial^2 u_i}{\partial x^2}, \quad (x,t) \in \Omega_i, \quad i = 1,2 \tag{14}$$

where k_1, k_2 are thermal diffusivity parameters. For simplicity, here we assume that $\Omega = \{(x,t) : (0,L) \times (0,T]\}$ is a rectangular domain which is subdivided by a latent moving interface $s(t)$ that separates the two phases as

$$\Omega_1 = \{(x,t) \in \Omega : 0 < x < s(t), t \in (0,T]\} \tag{15}$$

$$\Omega_2 = \{(x,t) \in \Omega : s(t) < x < L, t \in (0,T]\} \tag{16}$$

Energy balance implies the following initial and boundary Stefan conditions

$$u_1(s(t),t) = u_2(s(t),t) = u^*, \quad t \in [0,1] \tag{17}$$

$$s'(t) = \alpha_1 \frac{\partial u_1}{\partial x}(s(t),t) + \alpha_2 \frac{\partial u_2}{\partial x}(s(t),t), \quad t \in [0,1] \tag{18}$$

$$s(0) = s_0 \tag{19}$$

To illustrate the main capabilities of PINNs under this multi-physics setting, here we consider a prototype inverse one-dimensional two-phase Stefan problem, in a computational domain $\Omega = [0,2] \times [0,1]$ with given coefficients $k_1 = 2$, $k_2 = 1$, $\alpha_1 = -2$, $\alpha_2 = 1$, and $s_0 = 1/2$. Under this setting, one can fabricate a benchmark for which the exact solution for the temperature distribution and the moving boundary can be analytically derived as

$$u_1(x,t) = 2(\exp((t + 1/2 - x)/2) - 1) \tag{20}$$

$$u_2(x,t) = \exp((t + 1/2 - x) - 1) \tag{21}$$

$$s(t) = t + 1/2 \tag{22}$$

Now, suppose that α_1, α_2 are known. Given some measurements of the temperature distribution inside the domain Ω, our goal is to predict the latent functions $\{u_1(x,t), u_2(x,t), s(t)\}$ satisfying the system of Eqs. (14)–(19). Moreover, we would also like to infer the unknown thermal diffusivities k_1 and k_2.

A training dataset for this benchmark can be generated by randomly sampling $M = 200$ measurement points inside the domain Ω and obtain corresponding data for u^i using Eq. (25), i.e., $\{(x^i_{\text{data}}, t^i_{\text{data}}), u^i\}_{i=1}^M$. It is worth emphasizing that for any given datapoint $\{(x^i_{\text{data}}, t^i_{\text{data}}), u^i)\}$, we do not know the corresponding equation that governs u^i during training.

Results. As illustrated in Fig. 11, we can employ a PINN model to represent the unknown interface $s(t)$ by a deep fully connected neural network (five layers, 100 hidden units, and tanh activations) $s_{\boldsymbol{\beta}}(t)$. Similarly, we represent temperature distributions $u_1(x,t)$ and $u_2(x,t)$ by another fully connected neural network (five layers, 100 hidden units, and tanh activations) with two outputs $u^{(1)}_{\boldsymbol{\theta}}(x,t)$ and $u^{(2)}_{\boldsymbol{\theta}}(x,t)$, i.e.,

$$[x,t] \xrightarrow{u_{\boldsymbol{\theta}}} [u^{(1)}_{\boldsymbol{\theta}}(x,t), u^{(2)}_{\boldsymbol{\theta}}(x,t)] \tag{23}$$

In this way, the final predicted solution in the whole domain Ω can be given by

$$u_{\boldsymbol{\theta}}(x,t) = u^{(1)}_{\boldsymbol{\theta}}(x,t)\mathbf{1}_{(0,s_{\boldsymbol{\beta}}(t))}(x) + u^{(2)}_{\boldsymbol{\theta}}(x,t)\mathbf{1}_{(s_{\boldsymbol{\beta}}(t),1)}(x) \tag{24}$$

where $\mathbf{1}_{(0,s_{\boldsymbol{\beta}}(t))}(x)$ denotes an indicator function taking a value of one if $x \in [0, s_{\boldsymbol{\beta}}(t)]$ and zero otherwise. Then, the exact temperature solution in the whole domain Ω can be expressed by

$$u_{\text{exact}}(x,t) = u_1(x,t)\mathbf{1}_{(0,s(t))}(x) + u_2(x,t)\mathbf{1}_{(s(t),1)}(x) \tag{25}$$

The parameters $(\boldsymbol{\theta}, \boldsymbol{\beta})$ can be learned by minimizing the following mean squared error loss

$$\mathcal{L} = \sum_{k=1}^{2} [\mathcal{L}_r^{(k)} + \mathcal{L}_{s_{bc}}^{(k)}] + \mathcal{L}_{s_{Nc}} + \mathcal{L}_{s_0} + \mathcal{L}_{\text{data}} \tag{26}$$

where

$$\mathcal{L}_r^{(k)} = \frac{1}{N}\sum_{i=1}^{N} |\frac{\partial u^{(k)}_{\boldsymbol{\theta}}}{\partial t}(x^i,t^i) - \lambda_k \frac{\partial^2 u^{(k)}_{\boldsymbol{\theta}}}{\partial x^2}(x^i,t^i)|^2, \quad k = 1,2 \tag{27}$$

$$\mathcal{L}_{s_{bc}}^{(k)} = \frac{1}{N}\sum_{i=1}^{N} |u^{(k)}_{\boldsymbol{\theta}}\left(s_{\boldsymbol{\beta}}(t^i), t^i\right)|^2, \quad k = 1,2 \tag{28}$$

$$\mathcal{L}_{s_{Nc}} = \frac{1}{N}\sum_{i=1}^{N} |2\frac{\partial u^{(1)}_{\boldsymbol{\theta}}}{\partial x}\left(s_{\boldsymbol{\beta}}(t^i), t^i\right) - \frac{\partial u^{(2)}_{\boldsymbol{\theta}}}{\partial x}\left(s_{\boldsymbol{\beta}}(t^i), t^i\right) + \frac{ds_{\boldsymbol{\beta}}}{dt}(t^i)|^2 \tag{29}$$

$$\mathcal{L}_{s_0} = \frac{1}{N}\sum_{i=1}^{N} |s_{\boldsymbol{\beta}}(0) - s_0|^2 \tag{30}$$

$$\mathcal{L}_{\text{data}} = \frac{1}{M}\sum_{i=1}^{M} |u_{\boldsymbol{\theta}}(x^i_{\text{data}}, t^i_{\text{data}}) - u^i|^2 \tag{31}$$

Here N denotes the batch size, $\{(x^i,t^i)\}_{i=1}^N$ are collocation points that are randomly sampled at each iteration of the gradient descent. In addition, λ_1, λ_2 are two extra trainable parameters. Since we know that the thermal diffusivity parameters cannot be negative, we initialize them at 0.1 and constrain them to remain positive during model training.

Figure 12 summarizes the predictions of the proposed PINN model, indicating an accurate reconstruction of both the latent temperature field, as well as the dynamic interface $s(t)$. Moreover, Fig. 13(b) shows the identified parameters k_1 and k_2. Here, we can observe a discrepancy between the true and identified parameters indicating that the PINN model described above fails to correctly identify the unknown thermal diffusivity parameters, even after

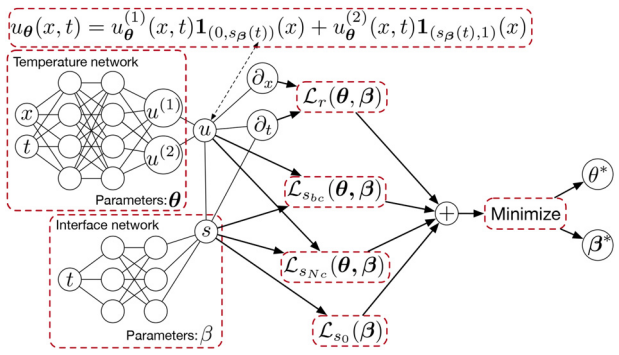


Fig. 11 Two-phase Stefan problem: PINN architecture for inferring the latent temperature fields $u_1(x,t)$, $u_2(x,t)$, and phase-transition interface $s(t)$, from scattered noisy observations of temperature

200,000 training iterations. In fact, we can observe that the two identified parameters do not change as training goes on. This observation suggests that our model seems to get stuck in some local minimum, and, as a result, fails to correctly recover the target parameter values. This indicates that the observed inaccuracy in the model's predictions is not due to insufficient training iterations, but relates to some issue pertaining to the model itself.

To resolve this issue, we proceed by applying the strategy of dynamic weights put forth by Wang et al. [58]. Specifically, we reformulate the loss function as

$$\mathcal{L} = \sum_{k=1}^{2}[\mathcal{L}_r^{(k)} + \mathcal{L}_{s_{bc}}^{(k)}] + \mathcal{L}_{s_{Nc}}) + \mathcal{L}_{s_0} + \lambda\mathcal{L}_{\text{data}} \tag{32}$$

where the weight parameter λ is adaptively updated during training by utilizing the back-propagated gradient statistics [58]. Specifically, the estimates of λ are computed by

$$\hat{\lambda}^{(k+1)} = \max_{\boldsymbol{\theta}}\{|\nabla_{\boldsymbol{\theta}}\mathcal{L}_r|\}/\overline{|\nabla_{\boldsymbol{\theta}}\lambda^{(k)}\mathcal{L}_{\text{data}}|} \tag{33}$$

where $\max_{\boldsymbol{\theta}}\{|\nabla_{\boldsymbol{\theta}}\mathcal{L}_r|\}$ is the maximum value attained by $|\nabla_{\boldsymbol{\theta}}\mathcal{L}_r|$, and $\overline{|\nabla_{\boldsymbol{\theta}}\lambda^{(k)}\mathcal{L}_{\text{data}}|}$ denotes the mean of $|\nabla_{\boldsymbol{\theta}}\lambda^{(k)}\mathcal{L}_{\text{data}}|$ over the parameter space $\boldsymbol{\theta}$. The weighting coefficients λ for the next iteration are updated using a moving average of the form

$$\lambda^{(k+1)} = (1-\alpha)\lambda^{(k)} + \alpha\hat{\lambda}^{(k+1)} \tag{34}$$

with $\alpha = 0.1$. As demonstrated in Ref. [58], this adaptive strategy can effectively mitigate pathologies arising in the training of PINNs due to stiffness in their gradient flow dynamics.

Figure 13 and Table 1 present comparisons of the identified parameters between the original PINN formulation of Raissi et al. [23], and the proposed PINN formulation with adaptive weights [58,59]. Notice that the PINN with adaptive weights not only converges to the exact parameters much faster, but also yields a significantly improved identification accuracy. In addition, we also

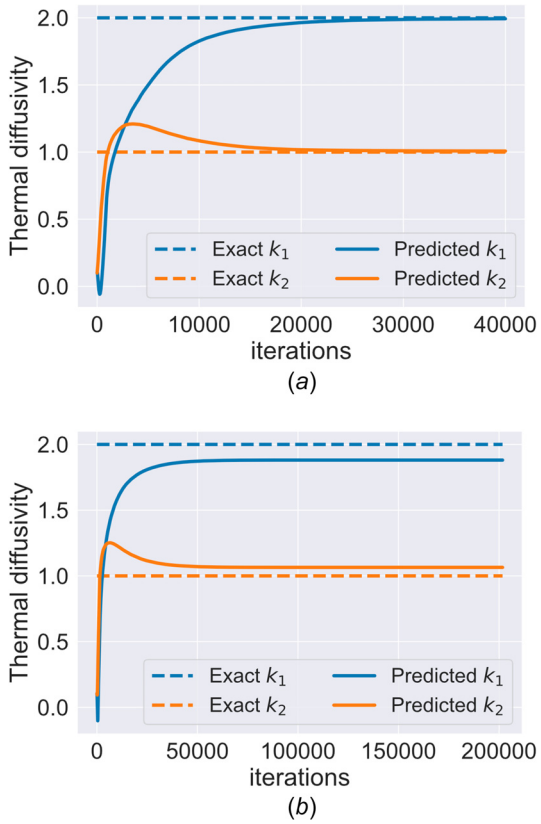


Fig. 13 Inverse one-dimensional two-phase Stefan problem type II with unknown parameters: convergence of the learned thermal diffusivity coefficients k_1 and k_2 using PINNs with (*a*) or without (*b*) the adaptive learning rate annealing put forth by Wang et al. [58]. (*a*) Inferred coefficients: $k_1 = 1.751$, $k_2 = 1.117$. (*b*) Inferred coefficients: $k_1 = 2.003$, $k_2 = 1.000$.

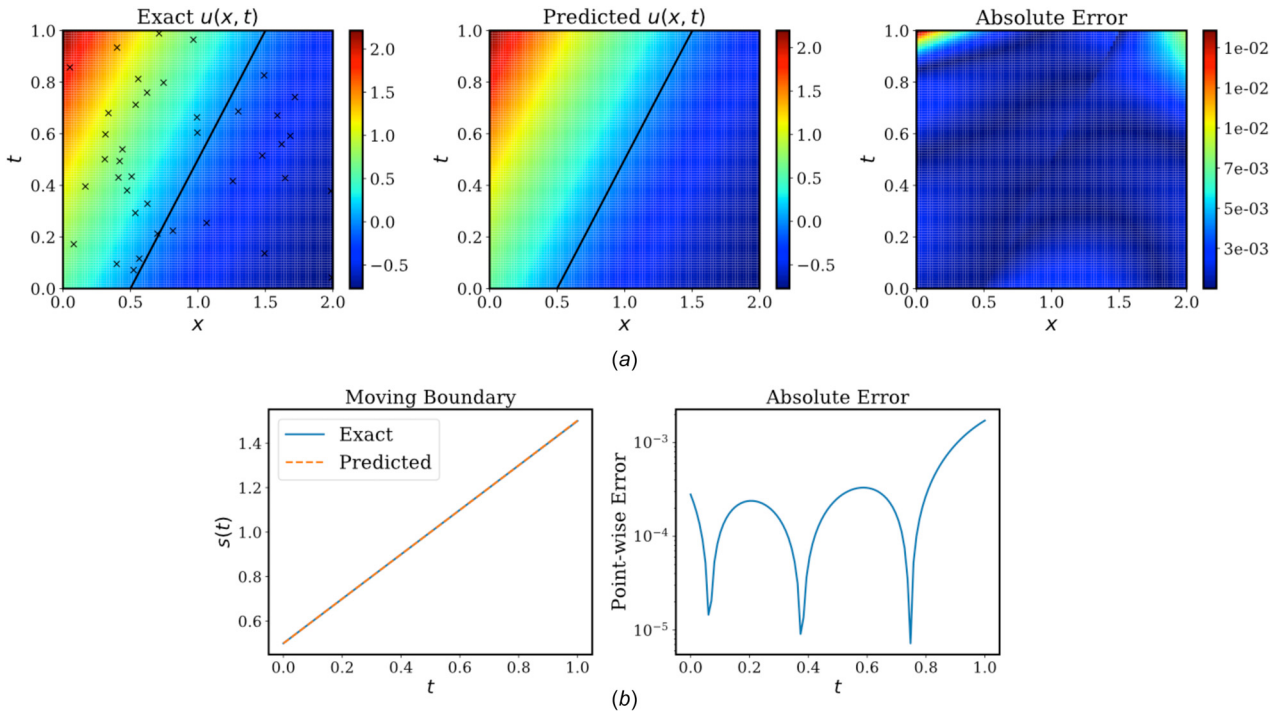


Fig. 12 Inverse two-phase Stefan problem: (*a*) exact temperature solution along with training data (×) versus the predicted solution. The relative L^2 error: 2.57×10^{-3}. (*b*) Left: comparison of the predicted and exact predicted moving boundary. Right: the absolute error between the exact and the predicted free boundary for $t \in [0,1]$. The L^2 error is 3.93×10^{-4}.

investigate the accuracy of the reconstructed temperature $u(x, t)$ and inferred interface $s(t)$ with respect to these two methods. A comparison of relative L^2 error in $u(x, t)$ and $s(t)$ between these two models is presented in Table 1 from which we can see that the dynamic weights approach improves the relative prediction error by about one order of magnitude. These figures and tables suggest that the weights in the loss function play an important role, and choosing appropriate weight coefficients can enhance the performance of PINNs by accelerating convergence and avoiding poor local minima.

5 Applications to Power Electronics

Dealing with the extreme heat fluxes in power electronics requires advanced cooling technologies. The target heat density levels can be $>1\,\text{kW/cm}^2$ and $>1\,\text{kW/cm}^3$ with a typical chip temperature rise 30 °C and maximum temperature difference across the chip footprint around 10 °C [60]. In the following, we present two recent works by ANSYS and NVIDIA in modeling heat transfer in power electronics using PINNs.

5.1 Heat Transfer in Electronic Chips. The thermal management in semiconductors is extremely important in the AI chip-package-systems, 5G networks and automotive. To this end, solving heat transfer problems on electronic chips is investigated by Central ML Team at ANSYS. The heat transfer from a chip involves heat conduction in solid phase, heat transfer from the chip to the surrounding fluid (air) and radiation. To test the effectiveness of using PINNs to solve heat equations, various canonical cases, such as 2D transient heat transfer with Dirichlet boundary condition, 2D transient heat transfer with source, as well as some practical engineering problems, such as 2D and three-dimensional (3D) heat transfer on a chip, chip tile temperature prediction, etc., have been investigated. The experimental results are shown in Fig. 14.

In Fig. 14(*a*), a 2D transient heat transfer problem with Dirichlet boundary conditions (temperatures on top, bottom, left, and right walls are 150, 300, 50, and 100 K, respectively) is solved and the comparison between numerical simulation results and PINN prediction is given. A 2D transient heat transfer case with heat source in the center is depicted in Fig. 14(*b*). Since the temperature profile along the line passing through the center of the square is of interest, temperature calculations from a numerical solver and PINN are compared along this line. It is worth noting that even though the residual points only cover a certain range in time axis (e.g., 0–20 s), the trained model can make predictions beyond the training range for extrapolation (e.g., 40 s) with reasonable accuracy (PINN prediction in orange curve, numerical results in blue curve).

As for practical engineering problems, PINNs have been used for solving 2D heat transfer on a chip, which is illustrated in Fig. 14(*c*). Discrete and drastically distinct power is applied on each tile of the chip and the final temperature distribution on the chip needs to be obtained. As shown in Fig. 14(*c*), PINNs not only display satisfactory prediction on the overall temperature distribution, but they also demonstrate adequate accuracy for the temperature profile on the line passing through the hottest spot on the chip (PINN prediction in orange curve, numerical results in blue curve).

Inspired by PINN, an Auto Encoder and Image Gradient (AEIG)-based approach has also been proposed by the Central ML Team at ANSYS team for solving 2D and 3D chip thermal analysis [22]. Figure 14(*d*-1) presents some examples of the 2D power maps, their corresponding ground truth, network prediction, and the prediction error. The proposed network shows acceptable agreement with numerical simulation with mean absolute percentage error (MAPE) of 0.4%. Figure 14(*d*-2) demonstrates some prediction examples by AEIG on 3D power maps and the MAPE is 0.14%.

Moreover, a chip tile-based temperature prediction problem is solved and exhibited in Fig. 14(*e*). A chip consists of arrays of tiles and the temperature distribution under the condition that power is only applied on a single tile of the chip is concerned. Since there could be variations about parameters such as x and y coordinates of the tile with power, size of the tile, and power magnitude, etc., a parameterized PINN is utilized where these parameters together with the independent variables of the PDE are taken as inputs to the network. The trained model would then be used to make inferences on any points (unseen combinations of the parameters) within the input space. Temperature predictions by the parameterized PINN for different power source locations, tile sizes with power, power magnitudes, are demonstrated in Figs. 14(*e*-1), 14(*e*-2), and 14(*e*-3), respectively.

5.2 SimNet for Heat Sink Design. To date, the published literature on PINNs has been able to demonstrate the forward solution of only simple problems when not using training data. The gradients, singularities, and discontinuities introduced by complex geometries or complex physics make the forward solution of real-world problems without training data extraordinarily challenging. NVIDIA's SimNet[3] is a toolkit for researchers and engineers with dual goals of being an extensible research platform as well as a solver for real-world and industrial problems. Figure 15 shows results of a conjugate heat transfer problem for the Nvidia's DGX-A100 NVSwitch heat sink whose fin geometry are variable. In heat sink design, the objective is to minimize the peak temperature that can be reached at the source chip while satisfying a maximum pressure drop constraint. This is necessary to meet the operating temperature requirements of the chip on which the heat sink is mounted for cooling. In this example, SimNet trains on a parametrized geometry with multiple design variables in a single training run thereby significantly accelerating design space exploration. In contrast, traditional solvers are limited to single geometry simulations. A forward solution of parameterized complex geometry with turbulent fluid flow between thinly spaced fins without training data makes this problem extremely challenging for the neural networks. Once the training is complete, several geometry, material or physical parameter combinations can be evaluated using inference as a postprocessing step, without solving the forward problem again. Such throughput enables more efficient design space exploration tasks for complex systems in science and engineering. The neural networks in this example are trained with ten variables (with nine of them being geometry parameters) with min, max, and median values in the range of these parameters. The OpenFOAM and commercial solver runs are on 12 CPU cores (DUAL 3.4 GHz Intel Xeon Gold x6128), and the SimNet runs are on 8 V100 GPUs (DGX1). It can be seen that SimNet can solve the problem faster than traditional solvers by several orders of magnitude.

Table 1 Inverse one-dimensional two-phase Stefan problem type II with unknown parameters: correct partial differential equations (PDEs) along with the identified one obtained using physics-informed neural networks with or without the adaptive learning rate annealing put forth by Wang et al. [58]

Correct PDE	$\frac{\partial u_1}{\partial t} - 2\frac{\partial^2 u_1}{\partial x^2} = 0$	$\frac{\partial u_1}{\partial t} - \frac{\partial^2 u_1}{\partial x^2} = 0$
Identified PDE (original)	$\frac{\partial u_1}{\partial t} - 1.712\frac{\partial^2 u_1}{\partial x^2} = 0$	$\frac{\partial u_1}{\partial t} - 1.137\frac{\partial^2 u_1}{\partial x^2} = 0$
Identified PDE (adaptive)	$\frac{\partial u_1}{\partial t} - 2.003\frac{\partial^2 u_1}{\partial x^2} = 0$	$\frac{\partial u_1}{\partial t} - 1.000\frac{\partial^2 u_1}{\partial x^2} = 0$

6 Summary and Outlook

Employing neural networks in heat-transfer applications could accelerate progress in heat-transfer enhancement, thermal design,

[3] http://developer.nvidia.com

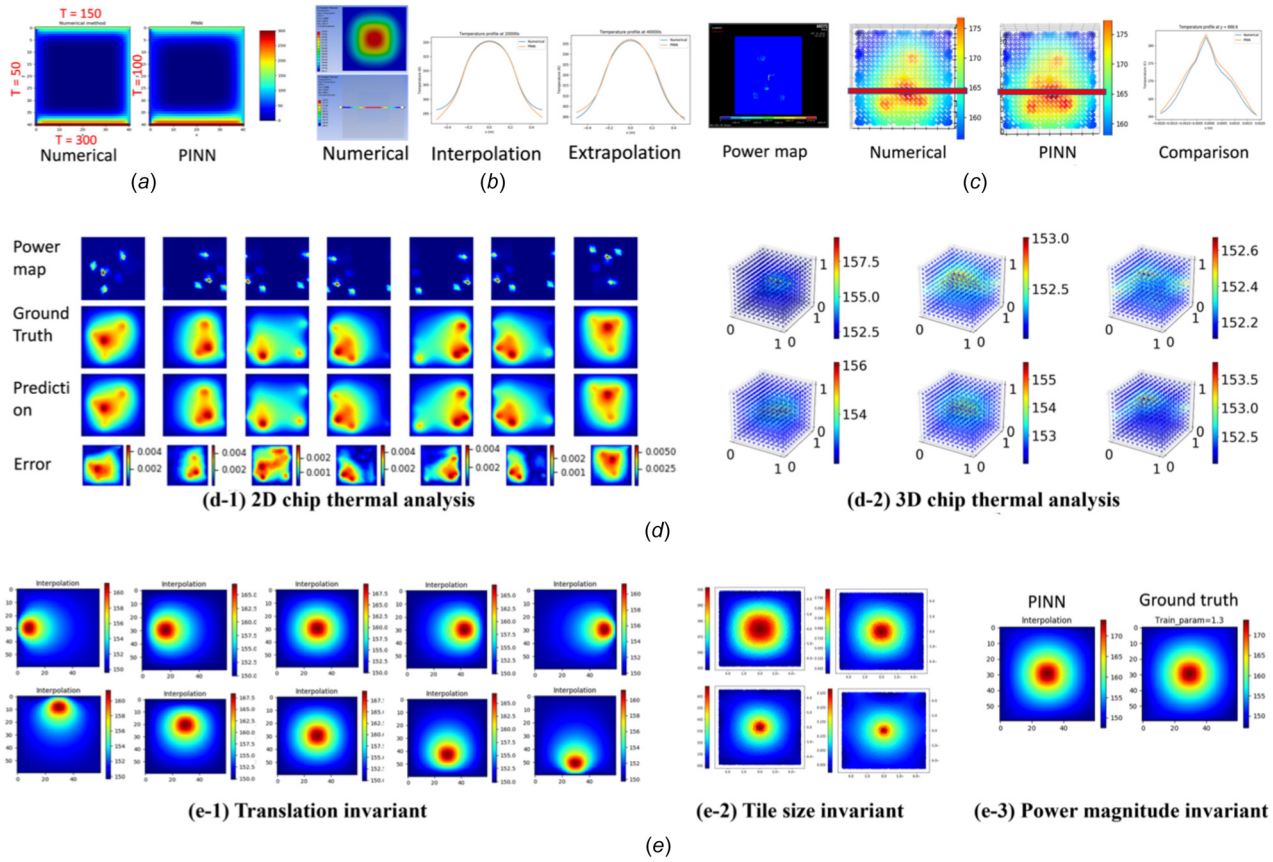
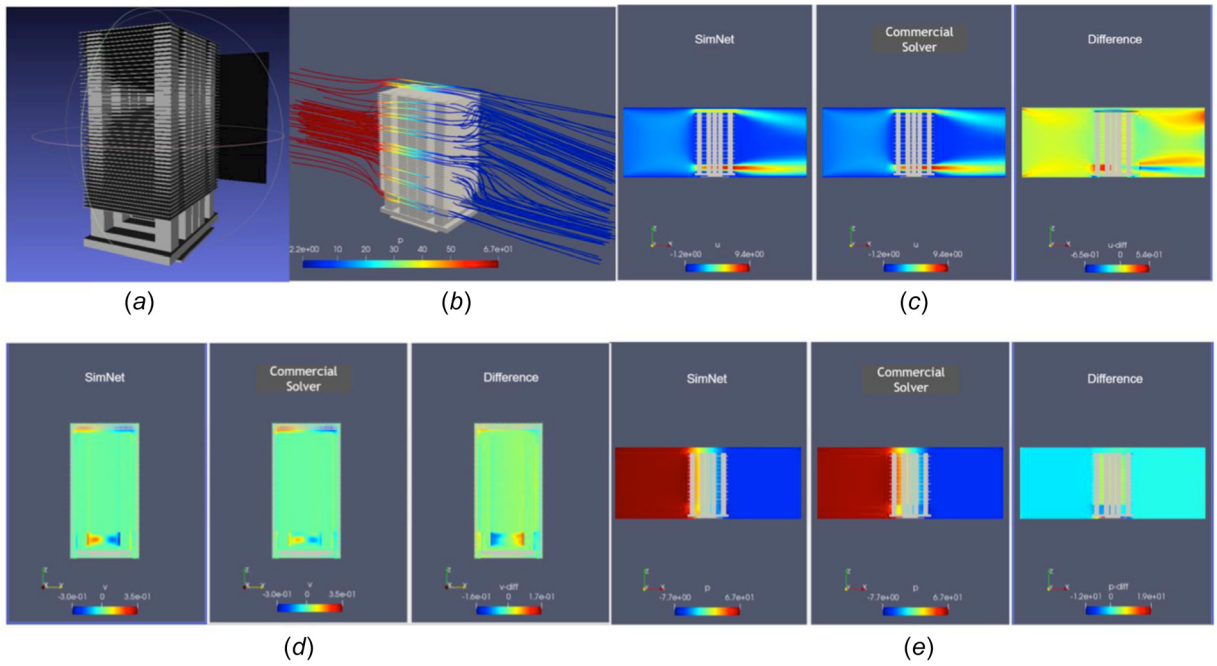


Fig. 14 Solving the heat equation on electronic chips. (*a*) 2D transient heat transfer problem with Dirichlet boundary conditions; (*b*) 2D transient heat transfer problem with heat source at the center of the square. Comparisons are made on the horizontal centerline within the training time zone (interpolation) and beyond the training time zone (extrapolation). (*c*) 2D heat transfer on a chip. Comparisons are performed on the red section lines passing through the hottest spot on chip. (*d*) Thermal analysis for 2D and 3D chips using PINN-inspired Auto-Encoder Image Gradient based approach [22]. (*e*) Temperature prediction on a chip with a single tile as power source by a parameterized PINN, for different power source locations, tile sizes with power, and power magnitudes. (Courtesy of the Central ML ANSYS team.)

and complex multiphase systems. However, the usual data-driven approach of neural networks is not appropriate due to the lack of big data and the various limitations on measuring velocity and temperature fields that may change rapidly in space and time. It is possible to measure temperature on part of a surface, in some cases, by employing the temperature-sensitive paint method (TSP) [61], while for higher temperatures phosphors can be used. The infrared cameras can also be used for measurements on a patch of a heated surface. Their resolutions varies from 320 × 320 to 1280 × 1024, they have good dynamic range at 14 bits, and uncertainties are ±2 °C for $T < 100$ °C, and ±2% of the reading for $T > 100$ °C. They can measure up to 600, 2000, and 3000 °C depending of which of the detectors is used. Within the flow, one can use the DPITV method [36], but further enhancements are required to improve its accuracy. In these methods, the response times are slow, of the order of milliseconds, so it will be necessary for critical dynamic applications to improve the response times down to 10 s of microseconds. The surface film TSP approach has a resolution that is equal to the size of the imaged pixel. For the bulk flow methods, the resolution would be of the order of the mean particle distance, which can be improved with standard interpolation methods but also with neural networks and especially PINNs that can fill gaps and provide high accuracy [20].

PINNs can play a catalytic role in bridging the gap between experimental and computational heat transfer. Using sparse measurements by employing the aforementioned multifidelity methods and encoding directly the conservation laws into the neural network architecture, inference of the velocity and temperature fields as well as unknown thermal boundary conditions or interfaces can be accurately obtained. PINNs offer a hybrid model, where we can use any data that is available at any locations and asynchronously in time while enforcing the governing equations using automatic differentiation at random points in the space–time domains without the need for elaborate and expensive mesh generation. Hence, PINNs are based on compact computer codes of 100 s of lines compared to 1000 s lines of code for traditional numerical solvers.

Moreover, PINNs can solve ill-posed inverse problems not possible with standard approaches, such as the omnipresent problem of unknown thermal boundary conditions. For decades, both experimental and computational research in heat transfer relied on idealized thermal boundary conditions of constant temperature or constant heat flux, neither of which is valid in practice. The pioneering experimental work of Robert Moffat [62], see, e.g., the Ph.D. thesis in Ref. [63], first addressed this issue by relaxing these assumptions, and we believe that PINNs is an effective approach to finally resolve this limitation. We demonstrated this possibility in this article by considering forced and mixed heat transfer with unknown thermal boundary conditions, having only a small number of temperature measurements on the heated cylinder or in the wake. Upon training the neural network, we can discover the entire boundary condition while simultaneously infer the velocity and temperature fields in the domain. The presented results indicate good performance of the method, e.g., for the case of constant temperature surface, using at most six measurements can correctly infer the temperature and Nusselt number profiles on



Turbulent Flow (Re=20,687)		
	Temperature	Pressure Drop
SimNet - Fourier Network	43.1 °C	3.56
Commercial Solver	43.5 °C	3.6
OpenFOAM	41.6 °C	4.58

Computational Times (10 parameters, 3 values per parameter)	
SimNet	1000 V100 GPU hrs.
Traditional Solver (OpenFOAM) 59,049 separate runs (26 wall hours on 12 CPU cores)	18.4M CPU core hrs.

Fig. 15 Solution of conjugate heat transfer problem on a complex parameterized geometry of a heat sink for NVSwitch in DGX-A100. (*a*) Geometry for a heat sink with fins and heat pipes to dissipate heat from GPU, (*b*) pressure color-coded flow streamlines and result comparisons between SimNet and CFD commercial code for (*c*) *U*-velocity, (*d*) *V*-velocity, and (*e*) pressure. (Courtesy of the NVIDIA team).

the boundary with less than 5% error. As the selection of the sensor location is very important for this inverse problem, we also proposed a method for active sensor placement. The residual of the heat transfer equation is considered as the criterion for selecting the sensor location. This metric is computed by the network, thus no prior knowledge is required. We assume that adding sensors at the location with highest value of residual can efficiently improve the performance of thermal boundary inference. Although the residual is related to the network structure and training process, we demonstrate the correlation between the residual and the posterior error of Nusselt number. As our test results show, this strategy can adaptively select the locations to place sensors while iteratively increase the prediction accuracy. Hence, PINNs can effectively guide the experimental work as well as bridging the gap between simulations and experiments as stated earlier by fusing any multifidelity measurements directly with the governing equations, which are encoded in the deep neural networks.

Here, we presented the basic version of PINNs but there have been many developments already, e.g., in employing domain decomposition in CPINN [64] and XPINN [65] that provide great flexibility with complex geometries but also in assigning a different neural network on each subdomain as well as parallel execution to accelerate inference. Another extension is the use of variational formulation in hp-VPINN [66]. Moreover, to quantify uncertainties one can use spectral expansions as in Ref. [67], generative models [68,69], or a Bayesian formulation as in Ref. [70]. Future work should address the possibility of multimodality/multifidelity measurements as inputs to PINNs, more efficient methods in quantifying uncertainties due to data as well as model uncertainties, tackling multidimensional multiphase problems, and producing proper benchmarks and data sets that can be used to further accelerate development of PINNs, especially in the industrial setting for thermal design, similar to the two applications we highlighted in this article.

Acknowledgment

The authors thank Professor Dana Dabiri of Washington University for providing the information on current measurement techniques. The authors thank the Nvidia team (Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Max Rietmann, Jose del Aguila Ferrandis, Wonmin Byeon, Zhiwei Fang, Sanjay Choudhry) and Mr. Jay Pathak and Dr. Haiyang He of Ansys for providing their results for Sec. 5 of the article.

Funding Data

- PhILMs (DOE DE-SC0019453).
- MURI/OSD (FA9550-20-1-0358).
- DOE grant (DE-SC0019116).

- AFOSR grant (FA9550-20-1-0060).
- DOE-ARPA grant (DE-AR0001201).

Nomenclature

ANN = artificial neural network
CFD = computational fluid mechanics
CNN = convolutional neural network
DPIT = digital particle image thermometry
PINN = physics-informed neural network
PDE = partial differential equation
TSP = temperature-sensitive paint

Physics

k = thermal diffusivity parameter
p = pressure
Pe = Péclet number
Ra = Rayleigh number
Re = Reynolds number
Ri = Richardson number
s = moving surface
t = temporal coordinate
u = velocity in x-direction
v = velocity in y-direction
x = spatial coordinate (x, y)
θ = temperature

Neural Network

b = neuron bias
N = number of training points
w = neuron weight
x_i = neuron input
y_j = neuron output
α = neuron amplitude for adaptive activation function
β = all parameters in the first neural network (Sec. 4)
θ = all parameters in the second neural network (Sec. 4)
λ = weighting coefficient in loss function
σ = activation function
$\mathcal{L}$ = loss function

References

[1] Jambunathan, K., Hartle, S., Ashforth-Frost, S., and Fontama, V., 1996, "Evaluating Convective Heat Transfer Coefficients Using Neural Networks," Int. J. Heat Mass Transfer, **39**(11), pp. 2329–2332.
[2] Liu, Y., Dinh, N., Sato, Y., and Niceno, B., 2018, "Data-Driven Modeling for Boiling Heat Transfer: Using Deep Neural Networks and High-Fidelity Simulation Results," Appl. Therm. Eng., **144**, pp. 305–320.
[3] Kwon, B., Ejaz, F., and Hwang, L. K., 2020, "Machine Learning for Heat Transfer Correlations," Int. Commun. Heat Mass Transfer, **116**, p. 104694.
[4] Tamaddon-Jahromi, H. R., Chakshu, N. K., Sazonov, I., Evans, L. M., Thomas, H., and Nithiarasu, P., 2020, "Data-Driven Inverse Modelling Through Neural Network (Deep Learning) and Computational Heat Transfer," Comput. Methods Appl. Mech., **369**, p. 113217.
[5] Fonda, E., Pandey, A., Schumacher, J., and Sreenivasan, K. R., 2019, "Deep Learning in Turbulent Convection Networks," Proc. Natl. Acad. Sci., **116**(18), pp. 8667–8672.
[6] Li, Y., Wang, H., and Deng, X., 2019, "Image-Based Reconstruction for a 3D-PFHS Heat Transfer Problem by ReconNN," Int. J. Heat Mass Transfer, **134**, pp. 656–667.
[7] Edalatifar, M., Tavakoli, M. B., Ghalambaz, M., and Setoudeh, F., 2020, "Using Deep Learning to Learn Physics of Conduction Heat Transfer," J. Therm. Anal. Calorim., pp. 1–18.
[8] Kim, J., and Lee, C., 2020, "Prediction of Turbulent Heat Transfer Using Convolutional Neural Networks," J. Fluid Mech., **882**, p. A18.
[9] Ronneberger, O., Fischer, P., and Brox, T., 2015, "U-Net: Convolutional Networks for Biomedical Image Segmentation," International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, Berlin, pp. 234–241.
[10] Smith, R., and Dutta, S., 2021, "Conjugate Thermal Optimization With Unsupervised Machine Learning," ASME J. Heat Transfer, **143**(5), p. 052901.
[11] Beintema, G., Corbetta, A., Biferale, L., and Toschi, F., 2020, "Controlling Rayleigh-Bénard Convection Via Reinforcement Learning," J. Turbul., **21**(9–10), pp. 585–605.
[12] Hachem, E., Ghraieb, H., Viquerat, J., Larcher, A., and Meliga, P., 2020, "Deep Reinforcement Learning for the Control of Conjugate Heat Transfer With Application to Workpiece Cooling," preprint arXiv..2011.15035.
[13] Raissi, M., Wang, Z., Triantafyllou, M. S., and Karniadakis, G. E., 2019, "Deep Learning of Vortex-Induced Vibrations," J. Fluid Mech., **861**, pp. 119–137.
[14] Mao, Z., Jagtap, A. D., and Karniadakis, G. E., 2020, "Physics-Informed Neural Networks for High-Speed Flows," Comput. Methods Appl. Mech., **360**, p. 112789.
[15] Jin, X., Cai, S., Li, H., and Karniadakis, G. E., 2021, "NSFnets (Navier-Stokes Flow Nets): Physics-Informed Neural Networks for the Incompressible Navier-Stokes Equations," J. Comput. Phys., **426**, p. 109951.
[16] Hennigh, O., Narasimhan, S., Nabian, M. A., Subramaniam, A., Tangsali, K., Rietmann, M., Ferrandis, J. D A., Byeon, W., Fang, Z., and Choudhry, S., 2020, "NVIDIA SimNet™: an AI-Accelerated Multi-Physics Simulation Framework," arXiv preprint arXiv:2012.07938.
[17] Cai, S., Wang, Z., Chryssostomidis, C., and Karniadakis, G. E., 2020, "Heat Transfer Prediction With Unknown Thermal Boundary Conditions Using Physics-Informed Neural Networks," ASME Paper No. FEDSM2020-20159.
[18] Wang, T., Huang, Z., Sun, Z., and Xi, G., 2021, "Reconstruction of Natural Convection Within an Enclosure Using Deep Neural Network," Int. J. Heat Mass Transfer, **164**, p. 120626.
[19] Lucor, D., Agrawal, A., and Sergent, A., 2021, "Physics-Aware Deep Neural Networks for Metamodeling of Turbulent Natural Convection," preprint arXiv:2103.03565.
[20] Cai, S., Wang, Z., Fuest, F., Jeon, Y. J., Gray, C., and Karniadakis, G. E., 2021, "Flow Over an Espresso Cup: Inferring 3D Velocity and Pressure Fields From Tomographic Background Oriented Schlieren Videos Via Physics-Informed Neural Networks," J. Fluid Mech..
[21] Wang, S., and Perdikaris, P., 2021, "Deep Learning of Free Boundary and Stefan Problems," J. Comput. Phys., **428**, p. 109914.
[22] He, H., and Pathak, J., 2020, "An Unsupervised Learning Approach to Solving Heat Equations on Chip Based on Auto Encoder and Image Gradient.," arXiv preprint arXiv:2007.09684.
[23] Raissi, M., Perdikaris, P., and Karniadakis, G. E., 2019, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," J. Comput. Phys., **378**, pp. 686–707.
[24] Psichogios, D. C., and Ungar, L. H., 1992, "A Hybrid Neural Network-First Principles Approach to Process Modeling," AIChE J., **38**(10), pp. 1499–1511.
[25] Lagaris, I. E., Likas, A., and Fotiadis, D. I., 1998, "Artificial Neural Networks for Solving Ordinary and Partial Differential Equations," IEEE Trans. Neural Network, **9**(5), pp. 987–1000.
[26] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., and Kudlur, M., 2016, "Tensorflow: A System for Large-Scale Machine Learning," 12th {USENIX} Symposium on Operating Systems Design and Implementation {OSDI}, Vol. 16, pp. 265–283.
[27] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A., 2017, "Automatic Differentiation in Pytorch," epub.
[28] Incropera, F. P., Lavine, A. S., Bergman, T. L., and DeWitt, D. P., 2007, *Fundamentals of Heat and Mass Transfer*, Wiley, Hoboken, NJ.
[29] Bejan, A., and Kraus, A. D., 2003, *Heat Transfer Handbook*, Wiley, Hoboken, NJ.
[30] Wang, Z., Chalfant, G. E. K. J., Chryssostomidis, C., and Babaee, H., 2017, "High-Fidelity Modeling and Optimization of Conjugate Heat Transfer in Arrays of Heated Cables," IEEE Electric Ship Technologies Symposium (ESTS), Arlington, VA, Aug. 14–17, pp. 557–563.
[31] D'Elia, M., Perego, M., and Veneziani, A., 2012, "A Variational Data Assimilation Procedure for the Incompressible Navier-Stokes Equations in Hemodynamics," J. Sci. Comput., **52**(2), pp. 340–359.
[32] Ozisik, M. N., 2018, *Inverse Heat Transfer: Fundamentals and Applications*, Routledge, London, UK.
[33] Karniadakis, G. E., and Sherwin, S., 2005, *Spectral/hp Element Methods for Computational Fluid Dynamics*, 2nd ed., Oxford University Press, Oxford, UK.
[34] Glorot, X., and Bengio, Y., 2010, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, pp. 249–256.
[35] Kingma, D. P., and Ba, J., 2014, "Adam: A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980.
[36] Dabiri, D., 2009, "Digital Particle Image Thermometry/Velocimetry: A Review," Exp. Fluids, **46**(2), pp. 191–241.
[37] Jagtap, A. D., Kawaguchi, K., and Em Karniadakis, G., 2020, "Locally Adaptive Activation Functions With Slope Recovery for Deep and Physics-Informed Neural Networks," Proc. R. Soc. London, Ser. A, **476**(2239), p. 20200334.
[38] Stefan, J., 1891, "Über Die Theorie Der Eisbildung, Insbesondere Über Die Eisbildung im Polarmeere," Ann. Phys., **278**(2), pp. 269–286.
[39] Tarzia, D. A., 2000, "A Bibliography on Moving-Free Boundary Problems for the Heat-Diffusion Equation. The Stefan and Related Problems," Materials, **2**, pp. 1–297.
[40] Busse, F., 1978, "Non-Linear Properties of Thermal Convection," Rep. Prog. Phys., **41**(12), pp. 1929–1967.
[41] Tsai, M., and Kou, S., 1989, "Marangoni Convection in Weld Pools With a Free Surface," Int. J. Numer. Methods Fluids, **9**(12), pp. 1503–1516.
[42] Friedman, A., 2000, "Free Boundary Problems in Science and Technology," Not. AMS, **47**(8), pp. 854–861.

[43] Dupret, F., Nicodeme, P., Ryckmans, Y., Wouters, P., and Crochet, M., 1990, "Global Modelling of Heat Transfer in Crystal Growth Furnaces," Int. J. Heat Mass Transfer, **33**(9), pp. 1849–1871.
[44] Madejski, J., 1976, "Solidification of Droplets on a Cold Surface," Int. J. Heat Mass Transfer, **19**(9), pp. 1009–1013.
[45] Sackinger, P., Brown, R., and Derby, J., 1989, "A Finite Element Method for Analysis of Fluid Flow, Heat Transfer and Free Interfaces in Czochralski Crystal Growth," Int. J. Numer. Methods Fluids, **9**(4), pp. 453–492.
[46] Cahn, J. W., 1960, "Theory of Crystal Growth and Interface Motion in Crystalline Materials," Acta Metall., **8**(8), pp. 554–562.
[47] Tekriwal, P., and Mazumder, J., 1988, "Finite Element Analysis of Three-Dimensional Transient Heat Transfer in GMA Welding," Weld. J., **67**(7), pp. 150s–156 s.
[48] David, S., Babu, S., and Vitek, J., 2003, "Welding: Solidification and Microstructure," JOM, **55**(6), pp. 14–20.
[49] Friedman, A., and Hu, B., 1995, "Stefan Problem With Kinetic Condition Arising in Semiconductor Processing," Pitman Res. Notes Math. Ser., pp. 121–121.
[50] Duvant, G., and Lions, J. L., 2012, *Inequalities in Mechanics and Physics*, Vol. 219, Springer Science & Business Media, Berlin.
[51] Crowley, A., 1978, "Numerical Solution of Stefan Problems," Int. J. Heat Mass Transfer, **21**(2), pp. 215–219.
[52] Voller, V., 1987, "An Implicit Enthalpy Solution for Phase Change Problems: With Application to a Binary Alloy Solidification," Appl. Math. Model., **11**(2), pp. 110–116.
[53] Date, A., 1992, "Novel Strongly Implicit Enthalpy Formulation for Multidimensional Stefan Problems," Numer. Heat. Transfer B, **21**(2), pp. 231–251.
[54] Fix, G. J., 1982, "Phase Field Methods for Free Boundary Problems," epub.
[55] Mackenzie, J., and Robertson, M., 2002, "A Moving Mesh Method for the Solution of the One-Dimensional Phase-Field Equations," J. Comput. Phys., **181**(2), pp. 526–544.
[56] Chen, S., Merriman, B., Osher, S., and Smereka, P., 1997, "A Simple Level Set Method for Solving Stefan Problems," J. Comput. Phys., **135**(1), pp. 8–29.
[57] Osher, S., Fedkiw, R., and Piechor, K., 2004, "Level Set Methods and Dynamic Implicit Surfaces," Appl. Mech. Rev., **57**(3), pp. B15–B15.
[58] Wang, S., Teng, Y., and Perdikaris, P., 2020, "Understanding and Mitigating Gradient Pathologies in Physics-Informed Neural Networks," arXiv preprint arXiv:2001.04536.
[59] Wang, S., Yu, X., and Perdikaris, P., 2020, "When and Why Pinns Fail to Train: A Neural Tangent Kernel Perspective," arXiv preprint arXiv:2007.14527.
[60] Garimella, S. V., Persoons, T., Weibel, J. A., and Gektin, V., 2017, "Electronics Thermal Management in Information and Communications Technologies: Challenges and Future Directions," IEEE Trans. Compon. Packag. Manuf., **7**(8), pp. 1191–1205.
[61] Prasad, A., 2016, "A Detailed Uncertainty Analysis of Heat Transfer Experiments Using Temperature Sensitive Paint," Ph.D. thesis, Embry-Riddle Aeronautical University, Daytona Beach, FL.
[62] Moffat, R. J., 1990, "Some Experimental Methods for Heat Transfer Studies," Exp. Therm. Fluid Sci., **3**(1), pp. 14–32.
[63] Maciejewski, P., 1991, "Heat Transfer With Very High Free-Stream Turbulence," Ph.D. thesis.
[64] Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E., 2020, "Conservative Physics-Informed Neural Networks on Discrete Domains for Conservation Laws: Applications to Forward and Inverse Problems," Comput. Methods Appl. Mech., **365**, p. 113028.
[65] Jagtap, A. D., and Karniadakis, G. E., 2020, "Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations," Commun. Comput. Phys., **28**(5), pp. 2002–2041.
[66] Kharazmi, E., Zhang, Z., and Karniadakis, G. E., 2021, "hp-VPINNs: Variational Physics-Informed Neural Networks With Domain Decomposition," Comput. Methods Appl. Mech., **374**, p. 113547.
[67] Zhang, D., Lu, L., Guo, L., and Karniadakis, G. E., 2019, "Quantifying Total Uncertainty in Physics-Informed Neural Networks for Solving Forward and Inverse Stochastic Problems," J. Comput. Phys., **397**, p. 108850.
[68] Yang, Y., and Perdikaris, P., 2019, "Adversarial Uncertainty Quantification in Physics-Informed Neural Networks," J. Comput. Phys., **394**, pp. 136–152.
[69] Yang, Y., and Perdikaris, P., 2018, "Physics-Informed Deep Generative Models," arXiv preprint arXiv:1812.03511.
[70] Yang, L., Meng, X., and Karniadakis, G. E., 2021, "B-PINNs: Bayesian Physics-Informed Neural Networks for Forward and Inverse PDE Problems With Noisy Data," J. Comput. Phys., **425**, p. 109913.