Physics Informed Neural Networks for heat conduction with phase change

Bahae-Eddine Madir, Francky Luddens, Corentin Lothodé, Ionut Danaila

Physics Informed Neural Networks for heat conduction with phase change

Bahae-Eddine Madir[a], Francky Luddens[a,*], Corentin Lothodé[b], Ionut Danaila[a]

[a] *Univ Rouen Normandie, CNRS, Normandie Univ, LMRS UMR 6085, F-76000 Rouen, France*
[b] *Univ Angers, CNRS, LAREMA - UMR 6093, SFR MATHSTIC, F-49000 Angers, France*

Abstract

We study numerical algorithms to solve a specific Partial Differential Equation (PDE), namely the Stefan problem, using Physics Informed Neural Networks (PINNs). This problem describes the heat propagation in a liquid-solid phase change system. It implies a heat equation and a discontinuity at the interface where the phase change occurs. In the context of PINNs, this model leads to difficulties in the learning process, especially near the interface of phase change. We present different strategies that can be used in this context. We illustrate our results and compare with classical solvers for PDEs (finite differences).

Keywords: Neural Networks, Partial Differential Equations, Physics Informed Neural Networks, Computational Physics, Phase Change Materials.

1. Introduction

When modeling heat transfer problems, depending on the context, there are various formulations. For a solid material involving a liquid-solid phase change, the problem reduces to the Stefan problem, taking into account only the conduction heat transfer mechanism (natural convection is ignored). This problem was studied extensively by the mathematical community, resulting in many theoretical results (see [1] and references therein). In the context of numerical simulation, several frameworks were developed to solve this problem (see the review [2]).

In the context of Finite Element Method (FEM), different approaches were suggested, from enthalpy based methods (e.g. [3]) to front-tracking methods (e.g. [4]). For other numerical frameworks, similar approaches have been developed, see for example [5] for Finite Volumes Method (FVM) and [6] for Finite Differences (FD).

Recently, machine learning methods have experienced significant growth, largely due to the improvement of computing hardware, such as GPUs. Thanks to these advancements, machine learning methods have found applications in various scientific domains, including heat transfer problems [7, 8, 9, 10]. The majority of these applications relies on the supervised learning approach, for which large databases obtained from experiments or numerical simulations are required to train a model. One of the main challenges is the potential insufficiency of the data acquired through experiments. Additionally, high-fidelity

*Corresponding author.

numerical simulators are inherently slow, restricting their applicability in generating extensive numerical data-sets. Conversely, low-fidelity numerical simulators are generally faster, allowing for the creation of larger data-sets, albeit with a trade-off of reduced model accuracy. To tackle this problem, the physics-informed neural networks (PINNs) framework was developed, and subsequently it was applied to a various fluid mechanics problems [11, 12, 13, 14], as well as heat transfer problems [15, 16, 17, 18, 19]. Instead of solving partial differential equations (PDEs) using classical simulation tools to generate training data-sets, the governing PDEs can be used directly to train models. One key advantage of such an approach is the efficiency in data assimilation problems where the PDE, initial or boundary conditions are not well known, but a finite number of sparse measurements inside the domain of interest is available. PINNs are also efficient when dealing with a PDE with parametric setting or high-dimensional PDE [20, 21].

In this article, we study the applicability of PINNs, to solve the Stefan problem. The article is organized as follows. In Sec. 2, we describe Stefan's problem and the enthalpy formulation; we solve the problem using a finite difference method to generate a reference solution. In Sec. 3, we present a brief overview of the PINNs method and we use it to solve the problem for two different configurations. We study also the loss weighting effect on the model training. In Sec. 4, we present different strategies that can be used to improve the model accuracy. We conclude in Sec. 5 with a brief summary and outlook.

2. The one-dimensional Stefan problem

2.1. Heating of a semi-infinite material subject to phase change

The Stefan problem is defined on a semi-infinite 1d domain $x \geq 0$. The domain is filled with a pure material, subject to liquid-solid phase change. The dimensionless temperature and enthalpy are defined as:

$$\theta \leftarrow \frac{T - T_f}{\delta T}, \qquad H \leftarrow \frac{H}{\rho\, c\, \delta T},$$

where the temperature scale for melting and solidification is $\delta T = \max(T_h - T_f, T_f - T_c)$, with T_h, T_c, and T_f the hot, cold, and fusion temperatures, respectively. The constants c and ρ are the specific heat and the density (assumed to be the same in the liquid and the solid). The left wall, denoted by Γ_l is kept at a constant temperature θ_l, and the initial temperature at $t = 0$ is θ_r. The problem can be modeled as:

$$\begin{cases} \partial_t H - Fo\, \partial_{xx}\theta = 0, \quad t > 0,\ x > 0, \\ \theta(t, 0) = \theta_l, \\ \lim_{x \to +\infty} \theta(t, x) = \theta_r, \end{cases} \tag{2.1}$$

where $Fo = \alpha\, t_{\text{ref}}/x_{\text{ref}}^2$ is the Fourier number, with α, t_{ref} and x_{ref} denoting the thermal diffusivity of the material, the reference time and reference length, respectively. The dimensionless enthalpy H is defined as

$$H(t, x) = \begin{cases} \theta(t, x) & \text{if } \theta(t, x) \leq 0, \\ \theta(t, x) + \dfrac{1}{Ste} & \text{if } \theta(t, x) > 0. \end{cases} \tag{2.2}$$

The Stefan number is defined as $Ste = c\, \delta T/L$, with L denoting the latent heat.

In this setting, the liquid-solid interface is a point $\lambda(t)$, and its motion is prescribed by the Stefan condition:

$$\partial_x \theta_{\text{right}}(t, \lambda(t)) - \partial_x \theta_{\text{left}}(t, \lambda(t)) = \frac{1}{Fo\, Ste} \lambda'(t),$$

where θ_{left} (resp. θ_{right}) corresponds to $\theta_{\text{ex}}(t, x)$ for $x < \lambda(t)$ (resp. $x > \lambda(t)$).

This condition is embedded in the enthalpy formulation (2.1). Using the ansatz $\lambda(t) = 2\lambda_0\sqrt{tFo}$, for some constant λ_0, the exact solution of the problem is given by:

$$\theta_{\text{ex}}(t, x) = \begin{cases} \theta_l \left(1 - \operatorname{erf}(\lambda_0)^{-1} \operatorname{erf}\left(\frac{x}{2\sqrt{tFo}}\right)\right), & \text{if } x \leq \lambda(t), \\ \theta_r \left(1 - \operatorname{erfc}(\lambda_0)^{-1} \operatorname{erfc}\left(\frac{x}{2\sqrt{tFo}}\right)\right), & \text{otherwise.} \end{cases} \tag{2.3}$$

The Stefan condition is satisfied provided λ_0 is the solution of the nonlinear equation:

$$\lambda_0 - \frac{Ste}{\sqrt{\pi}} e^{-\lambda_0^2} \left(\frac{\theta_r}{\operatorname{erfc}(\lambda_0)} + \frac{\theta_l}{\operatorname{erf}(\lambda_0)}\right) = 0.$$

This solution is referred to as the *exact solution* and will be used to set a suitable boundary condition on a finite domain.

2.2. Restriction to a finite domain

Numerically, we do not compute the solution on a semi-infinite domain. If the right boundary is kept at the temperature θ_r, then the exact solution holds only for small times, for which the interface is far away from the boundary. In order to avoid using a large computational domain, we need to solve numerically the problem on a smaller domain, with a time-dependent Dirichlet condition that matches θ_{ex}.

We consider a unit domain $D = [0, 1]$. The left and right walls, denoted by Γ_l and Γ_r are kept at a temperature θ_l and $\theta_{\text{ex}}(t, 1)$ respectively. The evolution of the temperature is described by the following PDE:

$$\partial_t H(t, x) - Fo\, \partial_{xx} \theta(t, x) = 0, \quad t > 0, \quad x \in D, \tag{2.4}$$

$$\theta(t, 0) = \theta_l, \tag{2.5}$$

$$\theta(t, 1) = \theta_{\text{ex}}(t, 1). \tag{2.6}$$

Noting that (2.2) can be reformulated as $H = \theta + (1/Ste)\varphi(\theta)$, where φ is the Heaviside function representing the liquid fraction, the previous equation can be rewritten as

$$\partial_t \theta(t, x) - Fo\, \partial_{xx} \theta(t, x) = -\frac{1}{Ste} \partial_t \varphi(\theta). \tag{2.7}$$

This equation is solved numerically in [22, 23] using a regularized enthalpy. Let $\delta > 0$ be a regularization parameter, and define

$$\varphi_\delta(\theta) = \frac{1}{2}\left(1 + \tanh\left(\frac{\theta}{\delta}\right)\right). \tag{2.8}$$

Using the regularization φ_δ instead of φ, equation (2.7) reads

$$\partial_t \theta = r(\theta)\, \partial_{xx} \theta \qquad t \in [0.05, 1], \quad x \in [0, 1], \tag{2.9}$$

where

$$r(\theta) = Fo \left(1 + \frac{1}{Ste} \varphi'_\delta(\theta)\right)^{-1}.$$

2.3. Reference solution

As a regularization term has been used, θ_{ex} is no longer the solution of (2.9). In order to assess the accuracy of the neural network approach, we compute a reference solution of (2.9) using a finite differences scheme. Applying the Crank–Nicolson scheme [24] for time integration of (2.9) and centered finite differences for the space discretization, we obtain

$$\frac{\theta_i^{n+1} - \theta_i^n}{\Delta t} = \frac{1}{2}\left[r(\theta_i^{n+1}) \frac{\theta_{i+1}^{n+1} - 2\theta_i^{n+1} + \theta_{i-1}^{n+1}}{(\Delta x)^2} + r(\theta_i^n) \frac{\theta_{i+1}^n - 2\theta_i^n + \theta_{i-1}^n}{(\Delta x)^2} \right]. \tag{2.10}$$

Assuming that the solution θ^n is known, then θ^{n+1} is the solution of a non linear system of equations, which can be solved using the Newton–Raphson method [25]. Figure 1 shows the numerical convergence of the considered approach. By setting equal step size for time and space $h = \Delta t = \Delta x$, we obtain convergence of order $\mathcal{O}(h^2)$.

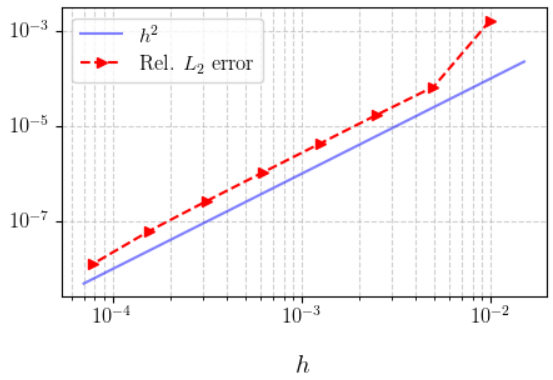

Figure 1: Convergence of the finite differences scheme used to solve (2.9). The L_2 error is calculated relative to a reference solution obtained with a step size $h_{\min} = 3.9 \times 10^{-5}$, i.e. Rel. L_2 error: $\|\theta_h - \theta_{h_{\min}}\|_2 / \|\theta_{h_{\min}}\|_2$.

3. Stefan problem with Physics-Informed Neural Networks

3.1. Physics-Informed Neural Networks

In this section, we provide a brief overview of the Physics-Informed Neural Networks (PINNs) [11], which is a method to approximate the solution to differential equations using neural networks.

A neural network with L layers (or $L-1$ hidden layers), can be represented as the composition of L parameterized functions $\ell^k(x, \Theta^k)$, where x and Θ^k are the input and a set of parameters for the $k-$th layer, respectively. A feed-forward neural network in particular, apply for each layer ℓ^k a linear and nonlinear transformations to the inputs i.e.

$$\forall k, \qquad \ell^k\left(x, \mathbf{W}^k; \mathbf{b}^k\right) = \Sigma^k\left(\mathbf{W}^k x + \mathbf{b}^k\right),$$

where $\mathbf{W}^k \in \mathbb{R}^{N_{k-1} \times N_k}$ and $\mathbf{b}^k \in \mathbb{R}^{N_k}$ are the weights and the biases which constitute the neural network parameters. N_k denotes the number of neurons in layer k and the Σ^k are nonlinear functions applied element-wisely, typically

$$\Sigma^k(x) = \begin{cases} \tanh(x), & \text{if} \quad 1 \leq k < L, \\ x, & \text{if} \quad k = L. \end{cases}$$

In the context of finding the solution u of a partial differential equation, a feedforward neural network can be considered as an approximate solution.

Consider PDEs taking the general form

$$\partial_t u + \mathcal{N}[u] = 0, \quad t \in [0, T], \quad x \in \Omega, \tag{3.1}$$

subject to initial and boundary conditions

$$u(0, x) = g(x), \quad x \in \Omega, \tag{3.2}$$
$$\mathcal{B}[u](t, x) = h(t, x), \quad t \in [0, T], \quad x \in \partial\Omega, \tag{3.3}$$

where $\mathcal{N}[\,\cdot\,]$ is a linear or nonlinear differential operator, and $\mathcal{B}[\,\cdot\,]$ is a boundary operator corresponding to Dirichlet, Neumann, Robin, or periodic boundary conditions.

By replacing the unknown solution $u(t, x)$ of (3.1), (3.2) and (3.3) with a neural network $u_\Theta(t, x)$, where Θ denotes all tunable parameters (weights and biases) of the network, the goal is to minimize the following composite loss function

$$\mathcal{L}(\Theta) = \omega_0\, \mathcal{L}_0(\Theta) + \omega_b\, \mathcal{L}_b(\Theta) + \omega_r\, \mathcal{L}_r(\Theta), \tag{3.4}$$

where $\mathcal{L}_r$, $\mathcal{L}_0$ and $\mathcal{L}_b$ are loss (residual) terms corresponding to the PDE term (3.1), the initial condition (3.2), and the boundary condition (3.3), respectively:

$$\mathcal{L}_r(\Theta) = \frac{1}{N_r} \sum_{k=1}^{N_r} \left| \frac{\partial u_\Theta}{\partial t}(t_r^k, x_r^k) + \mathcal{N}[u_\Theta](t_r^k, x_r^k) \right|^2, \tag{3.5}$$

$$\mathcal{L}_0(\Theta) = \frac{1}{N_0} \sum_{k=1}^{N_0} \left| u_\Theta(0, x_i^k) - g(x_i^k) \right|^2, \tag{3.6}$$

$$\mathcal{L}_b(\Theta) = \frac{1}{N_b} \sum_{k=1}^{N_b} \left| \mathcal{B}[u_\Theta](t_b^k, x_b^k) - h(t_b^k, x_b^k) \right|^2. \tag{3.7}$$

The sets $\{x_i^k, g(x_i^k)\}_{k=1}^{N_0}$, $\{t_b^k, x_b^k, h(t_b^k, x_b^k)\}_{k=1}^{N_b}$, and $\{t_r^k, x_r^k\}_{k=1}^{N_r}$ represent N_0 initial training points, N_b boundary training points, and N_r collocation points, respectively. They are randomly selected using low discrepancy sequence techniques (also known as a quasi-Monte Carlo sample) such as Latin Hypercube [26], Sobol sequence [27] or Halton [28] sequence.

It is worth noting that loss terms include gradients with respect to input variables. It is thus possible to compute these exactly using automatic differentiation [29] at any point in the domain, without the need for manual computation. Moreover, hyper-parameters ω_0, ω_b and ω_r enable the flexibility of assigning a different learning rate to each individual loss term in order to balance their interplay during model training. These weights may be user-specified or tuned automatically during training [30, 31].

3.2. Validation of the PINNs approach

We consider the equation (2.7) with $Fo = 0.01$, $Ste = 0.5$, a regularization $\delta = 0.05$ and a hot temperature $\theta_l = 1$. The initial condition is chosen to be the exact solution at the

initial time $t = 0.05$ (see (2.3)). The specific system is summarized as follows:

$$\partial_t \theta - 0.01\, \partial_{xx}\theta + 2\, \partial_t \left[\varphi_\delta(\theta)\right] = 0, \qquad t \in [0.05, 1], \quad x \in [0, 1] \tag{3.8a}$$

$$\theta(0.05, x) = \theta_{\text{ex}}(0.05, x), \tag{3.8b}$$

$$\theta(t, 0) = \theta_l, \tag{3.8c}$$

$$\theta(t, 1) = \theta_{\text{ex}}(t, 1). \tag{3.8d}$$

Following the original work of Raissi et al. [11], we represent the latent variable θ by a feed-forward neural network $\widehat{\theta}$ with 6 hidden layers and 20 neurons per hidden layer, we use the tanh as an activation function. In order to compute the loss functions, the Latin Hypercube Sampling method was adopted to generate $N_0 = 1024$ samples for the initial condition, $N_b = 256$ samples for the boundary conditions and $N_r = 10000$ collocation points in the spatio-temporal domain for the PDE term.

The model parameters are initialized using Xavier initialization [32], and the training procedure of the resulting PINN is done with full-batch gradient descent using the Adam optimizer [33] for $100k$ iterations with an exponential decay learning rate $t \mapsto \eta\, \gamma^{t/\kappa}$ where $\eta = 10^{-3}$, $\gamma = 0.9$, $\kappa = 8000$ and t is the training iteration. Table 1 summarizes the hyperparameters of both the physics-informed neural network and the training.

Hyperparameters	
Architecture	Two inputs (t, x), one output $\widehat{\theta}(t, x)$, six hidden layers of 20 neurons with the activation function tanh
Initialization	Xavier
Optimizer	Adam for $100k$ epoch
Learning rate	Exponential decay learning rate $t \mapsto \eta\, \gamma^{t/\kappa}$, $\eta = 10^{-3}$, $\gamma = 0.9$, $\kappa = 8000$
Training data	$N_0 = 1024$, $N_b = 256$, $N_r = 10000$ using LHC sampling

Table 1: Hyperparameters used to solve Eq. (3.8).

To evaluate the accuracy of the physics-informed neural network, we compare the model's prediction $\widehat{\theta}$ with the reference solution θ (given in sec. 2.3) at each time step, calculating the relative L_2 error $\|\widehat{\theta} - \theta\|_2 / \|\theta\|_2$ on a grid of 500×500 points in the spatio-temporal domain. The prediction θ is derived from averaging multiple simulations initialized independently, allowing us to consider the effect of initialization. Sensitivity to initialization will be demonstrated by the standard deviation of the simulation ensemble.

In this setting, we test the baseline PINNs approach introduced in [11], which sets $\omega_0 = \omega_b = \omega_r = 1$. From Fig. 3, we notice that the neural network captured accurately the solution of the problem. The relative L_2 error at the end of training in this case is $3.175 \times 10^{-3} \pm 0.0016$.

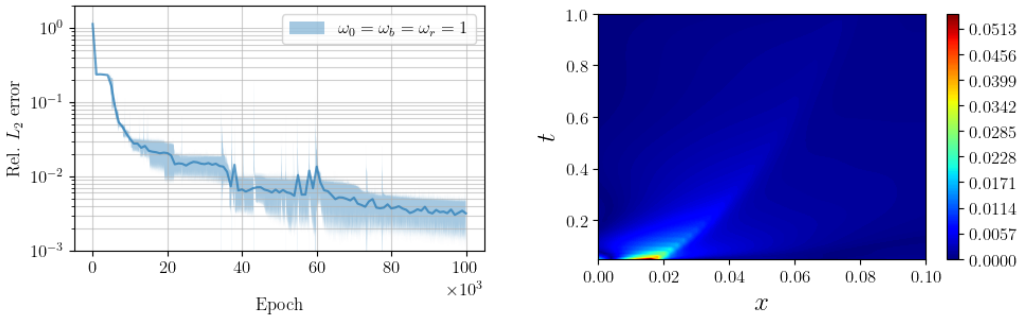

Figure 2: Problem (3.8). Left: relative L_2 error $\|\widehat{\theta} - \theta\|_2 / \|\theta\|_2$ during training. Right: absolute error $|\widehat{\theta} - \theta|$ at the end of training (i.e. at epoch $= 10^5$) .

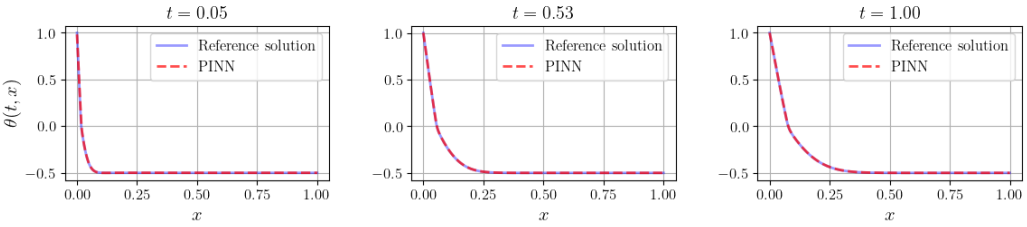

Figure 3: Solution of (3.8) at the end of training for $t = 0.05$, $t = 0.53$ and $t = 1$.

3.3. Influence of the Stefan number

In this section, we assess the ability of PINNs to capture the phase change with smaller values of parameter *Ste*. To this end, we change the Stefan number to the value $Ste = 0.005$. Then Eq. (3.8a) becomes:

$$\partial_t \theta - 0.01\, \partial_{xx}\theta + 200\, \partial_t \left[\varphi_\delta(\theta)\right] = 0, \qquad t \in [0.05, 1], \quad x \in [0, 1]. \tag{3.9}$$

In this scenario, the disparity between the coefficients of the time derivative $\partial_t \theta$ and the spatial derivative $\partial_{xx}\theta$ is greater than it was previously. As mentioned in [34], this kind of discrepancy could potentially pose challenges during training, as the complexity of the loss landscape increases, making the minimization of the loss function more challenging. For the model training, we maintain the same hyperparameters as previously. However, we introduce a supplementary adjustment by modifying the loss weights ω_0, ω_b, and ω_r. We perform three test cases: we use $\omega_0 = \omega_b = \omega_r = 1$ as previously. Secondly, we set $\omega_0 = 100$ and $\omega_b = \omega_r = 1$ intended to enforce the neural network to satisfy the initial condition. Finally, we use the dynamical weighting algorithm suggested in [30] to dynamically scale the weights of the loss function. The reweighting in this algorithm is performed so that the gradients of the loss terms of the initial and boundary conditions are approximately in the same range of values as the gradient of the PDE loss term. At each training step, e.g. iteration $(t+1)$, the estimates of ω_i and ω_b can be computed by:

$$\omega^{(t+1)} = (1-\alpha)\, \omega^{(t)} + \alpha\, \frac{\max_\theta\{|\nabla_\theta \mathcal{L}_r|\}}{\omega^{(t)} \overline{|\nabla_\theta \mathcal{L}|}}, \tag{3.10}$$

where $\max_\theta\{|\nabla_\theta \mathcal{L}_r|\}$ is the maximum value attained by $|\nabla_\theta \mathcal{L}_r|$ and $\overline{|\nabla_\theta \mathcal{L}|}$ denotes the mean of $|\nabla_\theta \mathcal{L}|$. The initial values of the dynamic weights are typically set to 1, and the parameter $\alpha = 0.6$ is used.

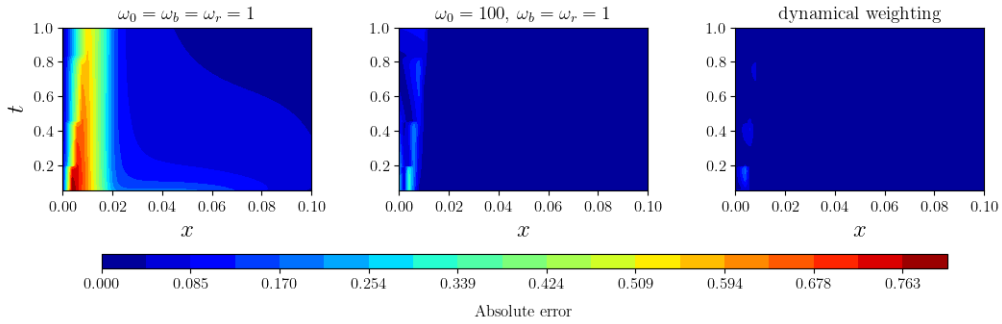

Figure 4: Absolute error $|\widehat{\theta} - \theta|$ for Eq. (3.9) at the end of training (i.e. at epoch $= 10^5$) for three choices of loss weights.

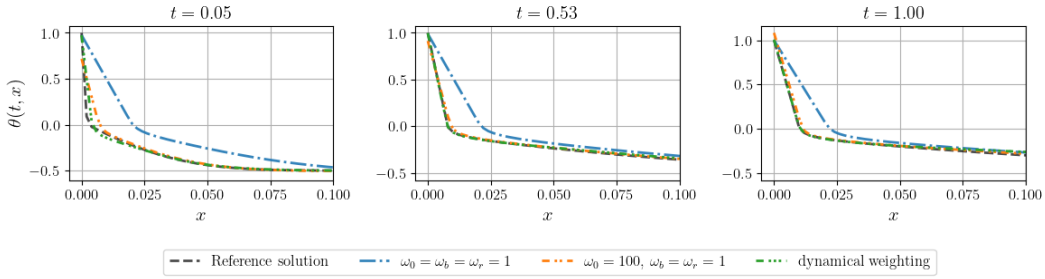

Figure 5: Solution of (3.9) at the end of training at $t = 0.05$, $t = 0.53$ and $t = 1$.

From Fig. 4, we can see that in all considered cases, the predominant portion of the error is located in the region near zero. This is primarily due to the gradual progression of the interface. Indeed, when we set $Ste = 0.005$ and $Fo = 0.01$, we deliberately introduce a movement of the interface with a steep gradient near zero. This type of behavior is difficult to model accurately for a neural network, as they are characterized by a high regularity.

In Fig. 5, where the solutions are plotted for $x \in [0, 0.1]$ at various time instants, we can observe distinct behavior for each case. When the weights are set to 1, the model is unable to learn the accurate solution. Furthermore, it seems that the predicted solution get stuck at some intermediate state and cannot be further refined. On the other hand, when $\omega_0 = 100$ is employed, the initial condition is learned much more effectively, resulting in a significantly more accurate solution at subsequent time instants. In the third scenario, where the learning rate annealing algorithm [30] is applied, we can observe that both the initial and boundary conditions are adhered to, leading to a more precise capture of the interface compared to the other two cases.

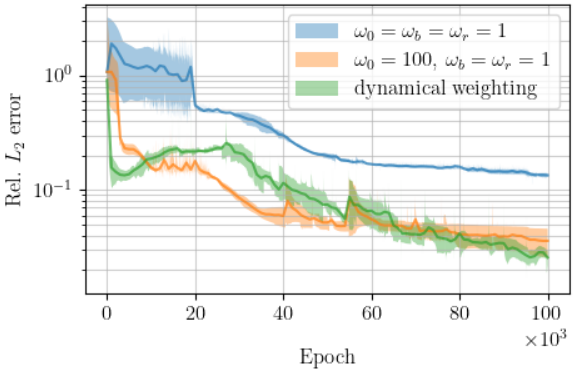

Figure 6: Relative L_2 error $\|\widehat{\theta} - \theta\|_2 / \|\theta\|_2$ during training for Eq. (3.9).

Figure 6, illustrates the significant role of the loss weights in the learning process. For instance, we notice the rapid decrease of the relative L_2 error in the first few thousand training epochs for both second (orange) and third (green) cases compared to the first one (blue). After the epoch $60k$, the relative L_2 error is barely changed for the first case while it is still considerably decreasing for the other two cases. Furthermore, the error for the third case became smaller than that of the second case. The relative L_2 error at the end of training, is $1.341 \times 10^{-1} \pm 0.005$ for the first case, $3.59 \times 10^{-2} \pm 0.01$ for the second case, and $2.565 \times 10^{-2} \pm 0.004$, when the dynamical weighting is performed with a frequency of 1000 iterations.

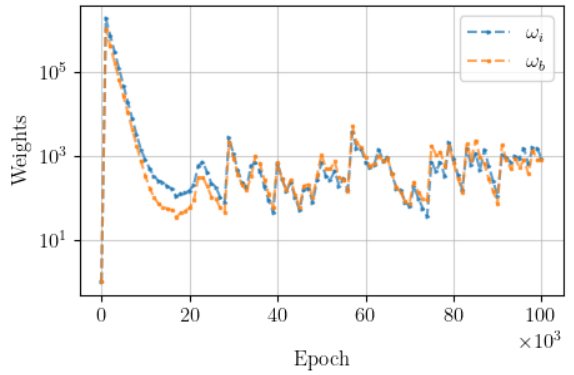

Figure 7: Values of the weights during training in the dynamical weighting case for Eq. (3.9).

Figure 7, shows that the weights are of the same order of magnitude. They reach very large values at the beginning of training to compensate for the dominance of the residual loss function. Over time, these weights gradually decrease and converge to an approximate value of 10^3.

4. Improvements of PINNs for the Stefan problem

In some cases, the PINNs method can struggle to capture an accurate solution. This difficulty may stem from the choice of various hyperparameters, such as the number of training data, the neural network architecture, the training parameters, etc. Despite careful consideration of these factors, there are instances where the method may still not perform well due to challenges in accurately capturing the underlying physics. In this section, we will explore some strategies aimed at refining the approximation of physics-informed neural networks.

4.1. Pointwise weighting using soft attention mask functions

As illustrated in the previous section, the weights within the loss function are a significant factor in the training process. In reality, these weights act as an adjustment to the learning rate during gradient descent for each component of the loss function. In [35] a pointwise weighting is suggested, to take in account the spatial imbalance of gradients; for every training (or collocation) point (t, x) we assign a weight $\omega(t, x)$. For the problem associated to Eqs. (3.1), (3.2) and (3.3), the structure of the loss function becomes:

$$\mathcal{L}(\Theta, \omega) = \mathcal{L}_0(\Theta, \omega_i) + \mathcal{L}_b(\Theta, \omega_b) + \mathcal{L}_r(\Theta, \omega_r), \tag{4.1}$$

where $\mathcal{L}_r$, $\mathcal{L}_0$ and $\mathcal{L}_b$ are loss terms corresponding to the PDE term (3.1), the initial condition (3.2), and the boundary condition (3.3), respectively:

$$\mathcal{L}_r(\Theta) = \frac{1}{N_r} \sum_{k=1}^{N_r} m_r(\omega_r^k) \left| \frac{\partial u_\Theta}{\partial t}(t_r^k, x_r^k) + \mathcal{N}[u_\Theta](t_r^k, x_r^k) \right|^2, \tag{4.2}$$

$$\mathcal{L}_0(\Theta) = \frac{1}{N_0} \sum_{k=1}^{N_0} m_i(\omega_i^k) \left| u_\Theta(0, x_i^k) - g(x_i^k) \right|^2, \tag{4.3}$$

$$\mathcal{L}_b(\Theta) = \frac{1}{N_b} \sum_{k=1}^{N_b} m_b(\omega_b^k) \left| \mathcal{B}[u_\Theta](t_b^k, x_b^k) - h(t_b^k, x_b^k) \right|^2, \tag{4.4}$$

where m_i, m_b and m_r are nonnegative, differentiable and strictly increasing functions, called mask functions. The weights $\{\omega_i^k\}_k$, $\{\omega_b^k\}_k$ and $\{\omega_r^k\}_k$ are updated by gradient descent side-by-side with the network parameters such that

$$\min_{\Theta} \max_{\omega_i, \omega_b, \omega_r} \mathcal{L}(\Theta, \omega_i, \omega_b, \omega_r) \tag{4.5}$$

is reached.

For the Stefan problem (3.9), the weighting is specifically applied to the initial and residual points. We employ mask functions (m_r, m_i, m_b) of the kind $w \mapsto \alpha(1+e^{-\beta(w-m)})^{-1}$, where $\alpha = 1000$, $\beta = 0.1$, and $m = 2$ for the initial condition points, while $\alpha = 1$, $\beta = 1$, and $m = 5$ are chosen for the residual points. The weights are initialized from a uniform distribution in $[0, 1]$ and are updated by gradient descent using the Adam optimizer with a learning rate $\eta_{\max} = 10^{-3}$ to maximize the loss function with respect to the weights w_i, w_r. To minimize the loss function with respect to the model parameters we use the Adam optimizer for $100k$ iteration with an exponential decay learning rate $t \mapsto \eta_{\min} \gamma^{t/\kappa}$ where $\eta_{\min} = 10^{-3}$, $\gamma = 0.9$, $\kappa = 5000$ and t is the training iteration. All other hyperparameters remain unchanged.

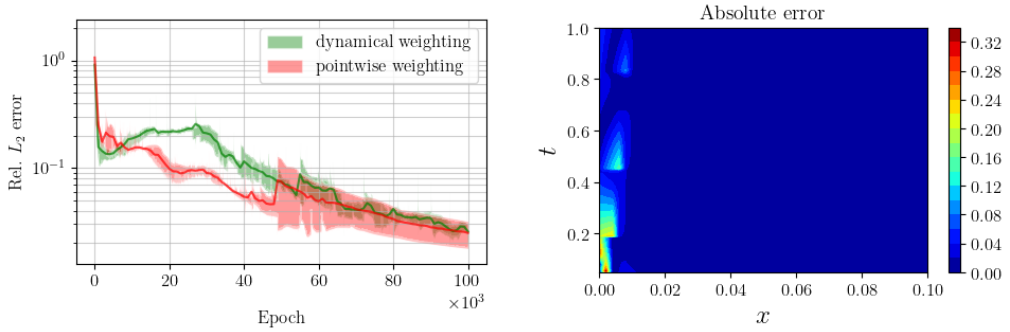

Figure 8: Problem (3.9). Relative L_2 error $\|\widehat{\theta} - \theta\|_2 / \|\theta\|_2$ during training (left). Absolute error $|\widehat{\theta} - \theta|$ at the end of training (i.e. at epoch $= 10^5$) (right).

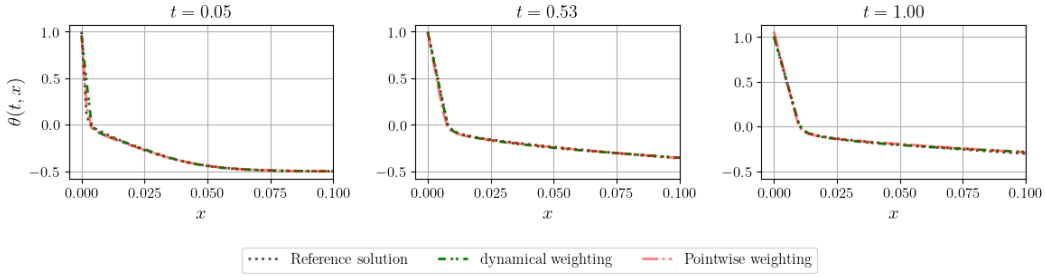

Figure 9: Solution of (3.9) at the end of training for the three cases in $t = 0.05$, $t = 0.53$ and $t = 1$.

Figure 8, reveals that during training, the initial L_2 error is lower with pointwise weighting than with dynamical weighting. However, beyond epoch $50k$, they converge to similar values (Fig. 9, shows the solutions at various time instants). The relative L_2 error at the end of training, is $2.484 \times 10^{-2} \pm 0.007$ for pointwise weighting and $2.565 \times 10^{-2} \pm 0.004$ the dynamical weighting.

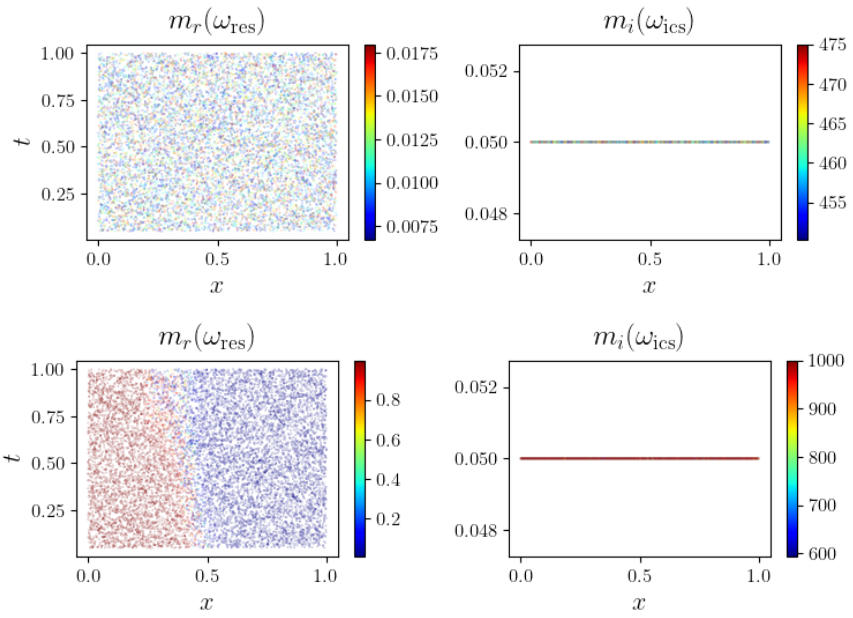

Figure 10: Problem (3.9). Values taken by the mask function for both the initial and residual points in the beginning of training (top) and at the end of training (bottom).

Figure 10, illustrates the multiplicative soft attention mask function values at the beginning and end of training. Initially, when the weights are initialized from a uniform distribution in the range $[0, 1]$, the values corresponding to the residual points are noticeably smaller than those for the initial points. This discrepancy is attributed to the hyperparameters α, β, and m, which emphasizes the learning for the initial condition, while constraining the dominance of the residual. At the end of training, we can observe that more attention is needed early in the solution, but not uniformly across the space variable.

To conclude, this approach can be an alternative to the dynamic weighting method presented in [30]. However, the selection of a mask function for each loss component requires prior knowledge about the challenges involved in fitting the problem.

4.2. Sequence learning in time

The conventional PINNs method developed by Raissi et al. [11] trains the neural network model to predict the solution across the entire space-time domain simultaneously. In certain cases, this can be difficult to learn. An alternative approach proposed in [36] involves training the model on sequences of the complete domain. Sequence learning approaches aim to simplify the training optimization problem, and have been proven to be effective [34].

In [36] Colby et al. considered the sequence $[0, k\Delta t] \times \Omega$, $k = 1, \ldots, N$ of the time-space domain $[0, T] \times \Omega$, where N is the number of subdomains and $\Delta t = T/N$ the step size. The model is trained firstly on $[0, \Delta t] \times \Omega$, then on $[0, 2\Delta t] \times \Omega$, and so on, gradually increasing the time span up to $[0, N\Delta t]$. To ensure that the solution is well-learned in each time subinterval, one can set a threshold or a maximum number of training iterations. Once the loss function value is below the threshold, or the training has reached the maximum number of iterations, the training procedure begins on the next time interval. In some

cases, if the loss function value is still very high after the maximum number of iterations, the time step size Δt may need to be reduced.

For our computational domain $D = [0.05, 1] \times [0, 1]$, we chose subdomains of the form $D_k = [0.05, 0.05 + k\Delta t] \times [0, 1]$ with uniform time step size $\Delta t = 0.05$. The number of collocation points $N_r^{(k)}$ used for D_k is 1000 if $k = 1$ and $N_r^{(k-1)} + 500$ otherwise. In order to prioritize the initial stages of learning, we determined the number of training iterations $N_{\text{it}}^{(k)}$ for each subdomain such that the product $N_{\text{it}}^{(k)} \times N_r^{(k)}$ is approximately equal to 10^8.

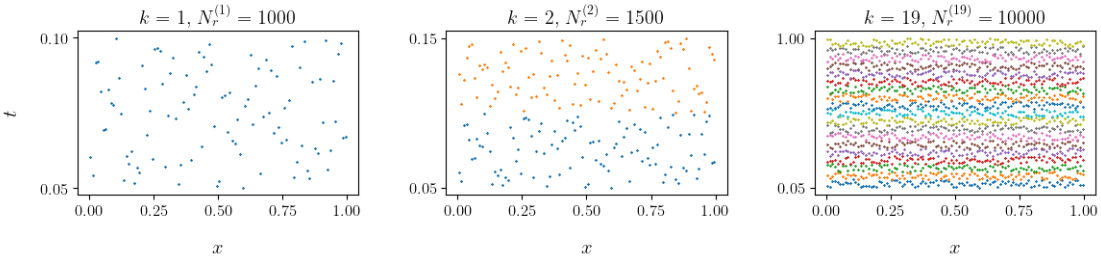

Figure 11: Example of collocation points for subdomains D_k. Note that the points used in subdomain D_{k-1} are also used in D_k.

Using this set of parameters, the total count of collocation points stands at 10000, which corresponds to the number used in the previous methods. Furthermore, the maximum number of iterations matches the minimum number of collocation points, ensuring a faster training process and more meaningful comparisons with the previous methods.

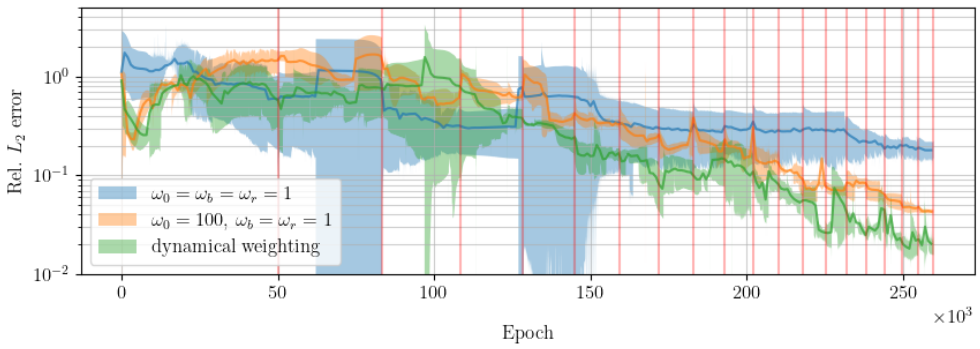

Figure 12: Problem (3.9). Relative L_2 error $\|\widehat{\theta} - \theta\|_2 / \|\theta\|_2$ during training. The red horizontal lines indicate the change of D_k sets.

Figure 12, shows the relative L_2 error during the training of the model. We notice that in the case when the coefficients of the loss function are set in a static way (i.e. $\omega_0 = \omega_b = \omega_r = 1$ or $\omega_0 = 100$, $\omega_b = \omega_r = 1$), the use of this approach did not improve the results. However in the case where the coefficients are set dynamically the error decreased. The relative L_2 error at the end of training, is $1.819 \times 10^{-1} \pm 0.036$ when $\omega_0 = \omega_b = \omega_r = 1$, $4.282 \times 10^{-2} \pm 0.001$ when $\omega_0 = 100$ and $\omega_b = \omega_r = 1$, and $1.887 \times 10^{-2} \pm 0.003$ when the weights are updated dynamically.

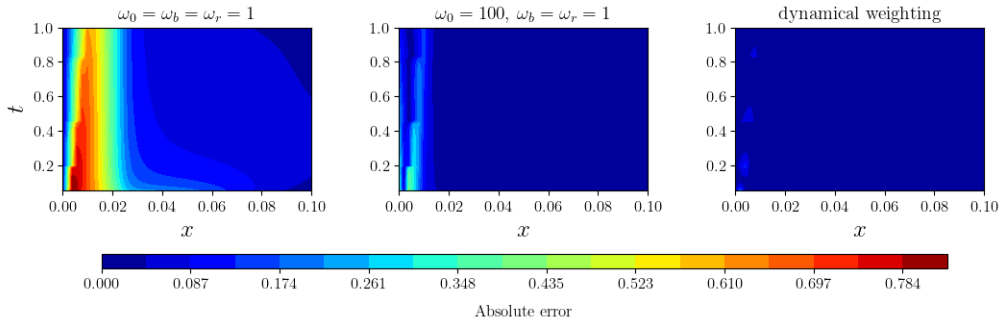

Figure 13: Absolute error $|\widehat{\theta} - \theta|$ for Eq. (3.9) when using sequence learning in time.

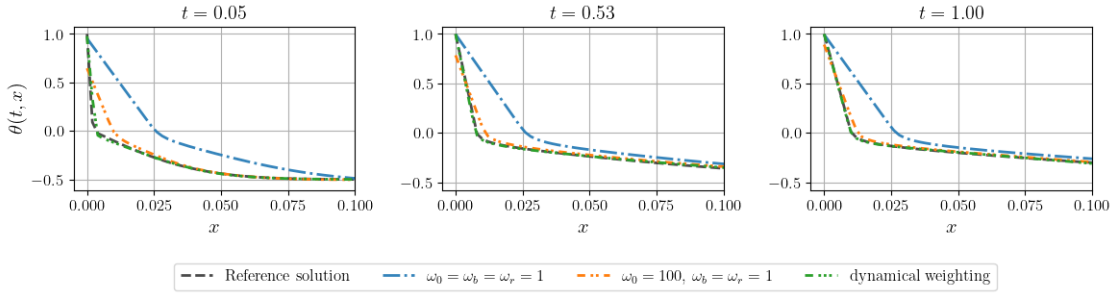

Figure 14: Solution of (3.9) when using sequence learning in time.

5. Conclusion

We focused on solving Stefan problem using the physics informed neural networks (PINNs) method. We explored various strategies to enhance the approximation capabilities of the PINNs, while maintaining a fixed neural network structure and a consistent number of training and collocation points.

By choosing a moderate value for the Stefan number (e.g. $Ste = 0.5$), the Stefan problem can be solved accurately using the PINNs method, even in the absence of loss function weighting. However, for smaller values (e.g. $Ste = 0.005$), the solution tends to exhibit less regularity, leading to increased complexity in capturing the interface. In this latter case, the exploration of new strategies becomes indispensable.

The classical approach employed in [11] does not account for the imbalance in gradients of the loss component. We have observed that incorporating weights in the loss function proved beneficial and enhanced the accuracy of the approximation. The drawback to manually selecting weights is the ongoing challenge of determining the suitable weights. Considering that, the learning rate annealing algorithm [30] has proven useful in identifying appropriate weights for the loss function. However, based on our experiences, these weights depend on the number of training and collocation points (N_i, N_b and N_r).
While weighting can be performed on a pointwise basis, determining the appropriate mask function is not straightforward. The approach of sequence learning in time holds promise for achieving higher accuracy since the optimization problem is confined to a smaller domain. However, a potential drawback, especially with more complex equations, is that

if the solution deviates within a time interval, subsequent intervals will propagate this error. It is important to learn the solution well on a time interval before progressing to the next one. This method may also be helpful when working on problems with larger time domains.

In general, a single approach can yield different results for various problems. In this work, we aimed to explore different methods to solve the Stefan problem. We did not examine the impact of network size, learning rate, or the number of training data on the accuracy of the approximation. Considering these parameters could lead to more precise solutions, and this will be one of the directions for our future research.

The Stefan problem studied in this paper is a classical phase change problem where the motion of the liquid phase, described by the Navier-Stokes equations and natural convection is ignored. In future studies, we will focus on a more realistic model of phase change problems, specifically the Navier-Stokes-Boussinesq equations.

Acknowledgement

This work has been supported by the regional project DATALAB. The authors also acknowledge financial support from the French ANR grant ANR-23-CE23-0020 (project PINNterfaces). Part of this work used computational resources provided by CRIANN (Centre Régional Informatique et d'Applications Numériques de Normandie).

References

[1] L Rubinstein. The Stefan problem: Comments on its present state. *IMA Journal of Applied Mathematics*, 24(3):259–277, 1979.

[2] Vaughan R Voller, CR Swaminathan, and Brian G Thomas. Fixed grid techniques for phase change problems: a review. *International Journal for Numerical Methods in Engineering*, 30(4):875–898, 1990.

[3] CM Elliott. Error analysis of the enthalpy method for the Stefan problem. *IMA Journal of Numerical Analysis*, 7(1):61–71, 1987.

[4] Chin Hsien Li. A finite-element front-tracking enthalpy method for Stefan problems. *IMA Journal of Numerical Analysis*, 3(1):87–107, 1983.

[5] Mohamad Muhieddine, Edouard Canot, and Ramiro March. Various approaches for solving problems in heat conduction with phase change. *International Journal on Finite Volumes*, page 19, 2009.

[6] Svetislav Savović and James Caldwell. Finite difference solution of one-dimensional Stefan problem with periodic boundary conditions. *International Journal of Heat and Mass Transfer*, 46(15):2911–2916, 2003.

[7] K Jambunathan, SL Hartle, S Ashforth-Frost, and VN Fontama. Evaluating convective heat transfer coefficients using neural networks. *International Journal of Heat and Mass Transfer*, 39(11):2329–2332, 1996.

[8] Yang Liu, Nam Dinh, Yohei Sato, and Bojan Niceno. Data-driven modeling for boiling heat transfer: using deep neural networks and high-fidelity simulation results. *Applied Thermal Engineering*, 144:305–320, 2018.

[9] Beomjin Kwon, Faizan Ejaz, and Leslie K Hwang. Machine learning for heat transfer correlations. *International Communications in Heat and Mass Transfer*, 116:104694, 2020.

[10] Hamid Reza Tamaddon-Jahromi, Neeraj Kavan Chakshu, Igor Sazonov, Llion M Evans, Hywel Thomas, and Perumal Nithiarasu. Data-driven inverse modelling through neural network (deep learning) and computational heat transfer. *Computer Methods in Applied Mechanics and Engineering*, 369:113217, 2020.

[11] 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. *Journal of Computational Physics*, 378:686–707, 2019.

[12] Maziar Raissi, Zhicheng Wang, Michael S Triantafyllou, and George Em Karniadakis. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137, 2019.

[13] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

[14] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

[15] Tongsheng Wang, Zhu Huang, Zhongguo Sun, and Guang Xi. Reconstruction of natural convection within an enclosure using deep neural network. *International Journal of Heat and Mass Transfer*, 164:120626, 2021.

[16] Didier Lucor, Atul Agrawal, and Anne Sergent. Physics-aware deep neural networks for surrogate modeling of turbulent natural convection. *arXiv preprint arXiv:2103.03565*, 2021.

[17] Shengze Cai, Zhicheng Wang, Frederik Fuest, Young Jin Jeon, Callum Gray, and George Em Karniadakis. Flow over an espresso cup: inferring 3-d velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks. *Journal of Fluid Mechanics*, 915:A102, 2021.

[18] Sifan Wang and Paris Perdikaris. Deep learning of free boundary and Stefan problems. *Journal of Computational Physics*, 428:109914, 2021.

[19] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-Informed Neural Networks for Heat Transfer Problems. *Journal of Heat Transfer*, 143(6):060801, 04 2021.

[20] Philipp Grohs, Fabian Hornung, Arnulf Jentzen, and Philippe Von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. *arXiv preprint arXiv:1809.02362*, 2018.

[21] Maziar Raissi. Forward–backward stochastic neural networks: deep learning of high-dimensional partial differential equations. In *Peter Carr Gedenkschrift: Research Advances in Mathematical Finance*, pages 637–655. World Scientific, 2024.

[22] Georges Sadaka, Aina Rakotondrandisa, Pierre-Henri Tournier, Francky Luddens, Corentin Lothodé, and Ionut Danaila. Parallel finite-element codes for the simulation of two-dimensional and three-dimensional solid–liquid phase-change systems with natural convection. *Computer Physics Communications*, 257:107492, 2020.

[23] Aina Rakotondrandisa, Georges Sadaka, and Ionut Danaila. A finite-element toolbox for the simulation of solid–liquid phase-change systems with natural convection. *Computer Physics Communications*, 253:107188, 2020.

[24] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50–67, 1947.

[25] Adi Ben-Israel. A Newton-Raphson method for the solution of systems of equations. *Journal of Mathematical Analysis and Applications*, 15(2):243–252, 1966.

[26] Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[27] Il'ya Meerovich Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

[28] John H Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, 1960.

[29] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18:1–43, 2018.

[30] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[31] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[32] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

[35] Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

[36] Colby L Wight and Jia Zhao. Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.