entropy

MDPI

Article

Improved Physics-Informed Neural Networks Combined with Small Sample Learning to Solve Two-Dimensional Stefan Problem

Jiawei Li, Wei Wu and Xinlong Feng *

College of Mathematics and System Sciences, Xinjiang University, Urumqi 830017, China; 1498975463@stu.xju.edu.cn (J.L.); wuwei837037@163.com (W.W.)
* Correspondence: fxlmath@xju.edu.cn; Tel.: +86-991-858-5505

Abstract: With the remarkable development of deep learning in the field of science, deep neural networks provide a new way to solve the Stefan problem. In this paper, deep neural networks combined with small sample learning and a general deep learning framework are proposed to solve the two-dimensional Stefan problem. In the case of adding less sample data, the model can be modified and the prediction accuracy can be improved. In addition, by solving the forward and inverse problems of the two-dimensional single-phase Stefan problem, it is verified that the improved method can accurately predict the solutions of the partial differential equations of the moving boundary and the dynamic interface.

Keywords: Stefan problem; deep neural networks; small sample learning; efficient calculation method

Citation: Li, J.; Wu, W.; Feng, X. Improved Physics-Informed Neural Networks Combined with Small Sample Learning to Solve Two-Dimensional Stefan Problem. *Entropy* **2023**, *25*, 675. https://doi.org/10.3390/e25040675

Academic Editor: Friedhelm Schwenker

Received: 12 March 2023
Revised: 10 April 2023
Accepted: 15 April 2023
Published: 18 April 2023

1. Introduction

For a fixed solution of a partial differential equation, it is usually restricted to a certain region. If this region varies with time, we call it the moving boundary problem. If part of the boundary of the fixed region is to be determined simultaneously with the solution of the fixed problem, we call it the free boundary problem, and the unknown boundary is called the free boundary. For free boundary problems, in addition to the usual fixed solution conditions, boundary conditions (Stefan conditions) must be added to the free boundary.

The Stefan problem is a class of heat conduction problems. It was first formulated by the Austrian physicist Joseph Stefan in the late 19th Century. The background of this problem is closely related to the industrial production during the industrial revolution of that time. In industrial production, many substances needed to be heat-treated, so the laws and methods of heat conduction needed to be studied to better control and utilize heat energy.

The difficulty of the Stefan problem is tracing the location of the interface, for example to model the process of change at the intersection of ice and water. Since the interface changes with time, these problems are also called free boundary problems. The study of free boundary problems has a wide practical background, such as plasma physics, percolation mechanics, and plasticity mechanics [1–6], which have presented various forms of constant and indeterminate free boundary problems. Furthermore, chemical vapor deposition [7], the vapor permeation of thermally cracked carbon in chemistry [8], tumor growth in medicine [9–13], the expansion propagation problem of biological populations [14–17], and the U.S. option pricing problem [18,19] also have free boundary problems. In fact, all free boundary problems are nonlinear problems, and it is important to solve them with the solution of the free boundary, which will be determined together with the solution of the fixed problem. Since these problems are closely related to practical applications, the efficient algorithmic implementation of free boundary problems is of great importance for scientific research and production practice.

Currently, a variety of numerical methods to the Stefan problems have emerged. The boundary integral method numerically solves the integral equation at a moving interface [20]. The interface tracing method explicitly represents the free boundary by using a fixed grid of points [1]. The immersion interface method [21] uses a fixed spatial grid for some physical quantities and a moving grid for the free boundary, and the information between the free boundary and the fixed grid is obtained by the immersion interface method. Segal [22] proposed an adaptive grid method for solving the free boundary problem, where the movement of the grid is determined by the control equations. Murray and Landis [23] compared the fixed-grid method with the adaptive-grid method. The adaptive-grid method can obtain the free boundary location more accurately, while the fixed-grid method can obtain certain physical quantities (temperature distribution over the whole solution area). More accurately, the enthalpy difference method [24] is an implicit method that represents the heat of the whole system by introducing an enthalpy function with a jump discontinuity at the free boundary, and this discontinuity helps to determine the location of the interface. The moving grid method [25] is a common method for solving free boundary problems, where the free boundary position is always fixed at the nth grid point, and the grid needs to be updated at each time step due to the movement of the free boundary. In recent years, the phase field method [26] has became a popular method based on the phase field function, which corresponds to a fixed constant in each phase and the interface region between the two values. This method considers a fuzzy boundary between the two phases, which is different from the classical Stefan problem, where the phase transition occurs in this interface region, where the thickness of the region is an artificially given parameter. The level-set approach [27] has also received increasing attention in recent years, by introducing a level-set function, which defines the interface position as a zero level set, obtained from the advection equation related to the velocity field, which varies considerably in different applications of the level-set approach. Sussman used the velocity of the fluid to model compressible two-phase flow [28], and Chen extended the interface moving velocity to the whole region by the advection equation in the solidification problem [27]. In contrast to the moving grid method, the level set method uses a fixed grid and avoids updating the grid at each time step. In contrast to the fixed-gridmethod, which finds the interface position at a fixed grid point, the phase-field method does not track the interface position precisely, and therefore, the discretization at the interface position is not as accurate as the fixed-grid method.

All the above methods have proven their high accuracy for some specific Stefan problems, and each has its own advantages and disadvantages. However, a general framework for solving the Stefan forward and inverse problems is still missing in all the methods at this stage.

In recent years, with the continuous development of neural networks, the use of neural networks to solve partial differential equations has gradually become popular [29–32]. Raissi et al. [33] proposed physics-informed neural networks for solving forward and inverse problems of partial differential equations, and deep-learning-based physics-informed neural networks have also been proposed for solving free boundary problems [34], as well as neural networks to solve the Stefan problems using a lattice-free grid-free automatic differentiation technique, which breaks the limitations of the above methods.

Deep learning models have achieved advanced results in solving various types of partial differential equations. However, the success of deep learning models relies heavily on a large amount of training data. In some specialized fields, the cost of data acquisition is very large. In addition, labeling samples requires much effort. Therefore, in recent years, a new learning approach, small sample learning, has gained popularity [35]. Small sample learning has been successfully applied to many new fields, such as: neural networks translation, target detection, etc. By using small sample learning, the accuracy of the model can be improved with few labeled data.

In this paper, we extended the recently emerging physics-informed neural networks framework [33] to solve the general Stefan problems. As we know, the original framework

of physics-informed neural networks does not deal well with free boundary problems with time variation. To achieve this goal, we propose and modified the neural network framework proposed by Wang [34]. The specific contributions of this paper can be summarized as follows: Firstly, we incorporated the idea of small sample learning into neural network training, i.e., a small sample loss is added to the loss function optimization in order to improve the training accuracy and correct the model. Secondly, we changed the loss function to cope with the outliers that may arise from free boundary shifts. Finally, we applied the proposed framework to irregular regions and irregular free boundaries to test its performance. In summary, the proposed method provides a new general framework for solving the Stefan problems.

This paper is structured as follows: Section 2 introduces the Stefan problem and its mathematical model. Section 3 introduces the knowledge of neural networks and the PINNs' improvement strategy. Section 4 verifies the accuracy and applicability of the proposed framework with numerical arithmetic examples. The conclusions and outlook are given in Section 5.

2. Model Issues

In this section, we introduce the mathematical model of the Stefan problem and the corresponding boundary conditions using a one-dimensional single-phase Stefan problem in the solidification or melting process as an example.

As shown in Figure 1, assume that a semi-infinite solid occupying $0 \leqslant x < \infty$ is in the process of solidification or melting. For any moment $t > 0$, the $0 \leqslant x < \infty$ region consists of a solid and a liquid. The liquid is located in the $0 \leqslant x < \alpha(t)$ region, and the solid is located in the $\alpha(t) < x < \infty$ region.

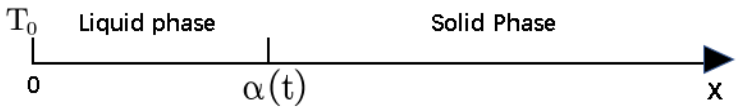

Figure 1. One-dimensional single-phase Stefan problem solidification or melting model.

If the volume change due to solidification or melting is not considered and the region $0 \leqslant x < \alpha(t)$ is considered, the temperature $u(x,t)$ satisfies the classical diffusion equation over the region:

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2}, \ x \in (0, \alpha(t)), \ t \in (0,T), \tag{1}$$

and for the initial and boundary conditions:

$$u(x,0) = u_0(x), \ x \in [0, \alpha(0)], \tag{2}$$
$$u(0,t) = h(t), \quad t \in [0,T]. \tag{3}$$

at the interface $\alpha(t)$, the following Stefan conditions need to be met:

$$\alpha(0) = \alpha_0, \tag{4}$$
$$u(\alpha(t),t) = 0, \quad t \in [0,T], \tag{5}$$
$$\frac{\partial u}{\partial x}(\alpha(t),t) = g(t), \ t \in [0,T], \tag{6}$$

where $\alpha(t)$ denotes the free boundary, (4) denotes the initial position of the free boundary, and (5) denotes the temperature at the time of freezing. For the forward problem, in thermal physics, it is the simultaneous solution of the temperature distribution and the free boundary for which various parameters are known. Each Stefan problem corresponds to a class of inverse problems, and similar to other mathematical physics inverse problems, the

inverse Stefan problem is not definite, while the uniqueness and stability of the solution are not always guaranteed.

3. Numerical Methods

In this section, the main methods and ideas of feedforward neural networks and physics-informed neural networks for solving partial differential equations are reviewed, and a general model of improved physics-informed neural networks combined with small sample learning for solving the Stefan problem is proposed.

3.1. Feedforward Neural Networks

Mathematically, a neural network is a specific class of complex functions, the simplest of which is a feedforward neural network (FNN), which is also called a multilayer perceptron (MLP). Let $\mathcal{F}^P(\mathbf{x}) : R^{d_{in}} \to R^{d_{out}}$ be a P-layer feedforward neural networks with $P-1$ hidden layers; the number of neurons in the input layer is $N_0 = d_{in}$, and the number of neurons in the output layer is $N_P = d_{out}$. The network weight of the pth layer is $\mathrm{W}^p \in R^{N_p \times N_p - 1}$, and the network bias is $\mathrm{b}^p \in R^{N_p}$. If given the activation function σ, the feedforward neural network (FNN) can be expressed as

$$\begin{aligned} &\mathcal{F}^0\mathbf{x} = \mathbf{x} \in R^{\mathrm{d_{in}}}, \\ &\mathcal{F}^p\mathbf{x} = \sigma(\mathrm{W}^p\mathcal{F}^{p-1}(\mathbf{x}) + \mathrm{b}^p) \in R^{N_p}, \\ &\mathcal{F}^P\mathbf{x} = \mathrm{W}^P\mathcal{F}^{P-1}(\mathbf{x}) + \mathrm{b}^P \in R^{\mathrm{d_{out}}}. \end{aligned} \tag{7}$$

3.2. Physics-Informed Neural Networks

We considered the following parameterized partial differential equation:

$$\begin{aligned} F(x,t;\partial_x u, \partial_t u, \ldots;\lambda) = 0, \ (x,t) \in \Omega \times (0,T], \\ u(x,0) = g(x), \ \ x \in \Omega, \\ \boldsymbol{B}u(x,t) = h(x,t), \ (x,t) \in \partial\Omega \times [0,T]. \end{aligned} \tag{8}$$

where $x \in \Omega \in \mathrm{R}^n$, F denotes the residuals of the partial differential equation, F in $(\partial_x u, \partial_t u, \ldots)$ denotes the space–time differential operator, $\lambda = [\lambda_1, \lambda_2, \lambda_3 \ldots]$ represents the parameters of the partial differential equation, $u(x,t)$ is the solution of the partial differential equation, Ω denotes the solution region of the partial differential equation, $\partial\Omega$ is the boundary of Ω, and $\boldsymbol{B}$ represents any of the boundary operators, including Dirichlet, Neumann, Robin, and periodic.

PINNs [33] use a fully connected feedforward neural networks consisting of multiple hidden layers to approximate the solution of the partial differential equation with spatio-temporal coordinates $u(x,t)$ as the input. The neural network is displayed in Figure 2.

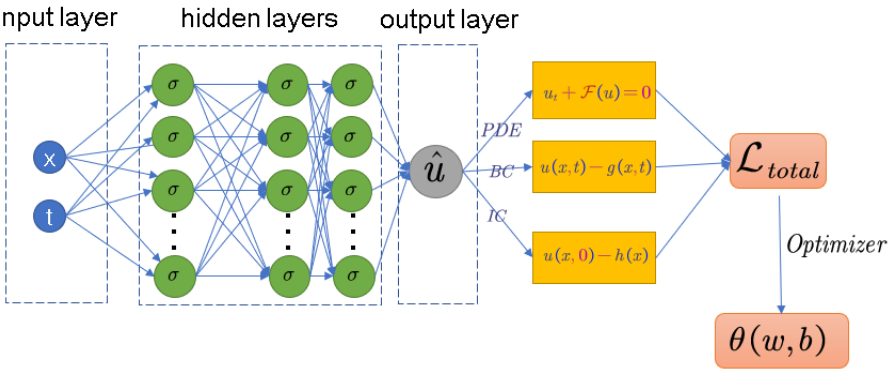

Figure 2. Physics-informed neural network structure diagram.

First, we establish the loss function based on the form of the partial differential equation and the initial margin value condition:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{f}}\mathcal{L}_{\text{f}} + \lambda_{\text{ic}}\mathcal{L}_{\text{ic}} + \lambda_{\text{bc}}\mathcal{L}_{\text{bc}}, \tag{9}$$

where

$$\begin{aligned}
\mathcal{L}_{\text{f}} &= \frac{1}{\text{N}_{\text{f}}}\sum_{\text{i}=1}^{\text{N}_{\text{f}}} \|F(x,t;\partial_x u, \partial_t u, \ldots;\lambda)\|_2^2, \\
\mathcal{L}_{\text{ic}} &= \frac{1}{\text{N}_{\text{ic}}}\sum_{\text{j}=1}^{\text{N}_{\text{ic}}} \left\| u\left(x_{\text{ic}}^{\text{j}}, 0\right) - g\left(x_{\text{ic}}^{\text{j}}\right)\right\|_2^2, \\
\mathcal{L}_{\text{bc}} &= \frac{1}{\text{N}_{\text{bc}}}\sum_{\text{k}=1}^{\text{N}_{\text{bc}}} \left\| \mathcal{B}\left(u\left(x_{\text{bc}}^{\text{k}}, t_{\text{bc}}^{\text{k}}\right)\right) - h\left(x_{\text{bc}}^{\text{k}}, t_{\text{bc}}^{\text{k}}\right)\right\|_2^2,
\end{aligned} \tag{10}$$

where $\mathcal{L}_{\text{f}}$, $\mathcal{L}_{\text{ic}}$, and $\mathcal{L}_{\text{bc}}$ denote the loss functions satisfying the control equations, initial conditions, and boundary conditions and λ_{f}, λ_{ic}, and λ_{bc} denote the weights of the loss functions $\mathcal{L}_{\text{f}}$, $\mathcal{L}_{\text{ic}}$, and $\mathcal{L}_{\text{bc}}$, respectively, where the derivatives involved are automatically differentiated by the neural networks.

After the loss function $\mathcal{L}_{\text{total}}$ is established, the optimal parameters $\theta^* = (\text{W}^*, \text{b}^*)$ of the neural networks are obtained by iterative updating with the objective of minimizing the loss function, considering that the loss function is nonlinear and nonconvex for θ^*, so a gradient-based optimizer is used to minimize the loss function, such as: Adagrad, AdaDelta, Adam, momentum, and RMSProp.

3.3. Neural Network Improvement Strategies for Stefan Problem

PINNs have been applied in various scientific and engineering fields to solve various complex physical problems. In the free boundary problem, the simple PINNs can no longer meet the requirements of the Stefan problem, and certain structures need to be changed accordingly. Therefore, we propose the following improvement strategies.

3.3.1. Neural Networks' Basic Structure

Recall that, for the Stefan problem, we aim to find both the temperature solution $u(x,t)$ and the moving boundary $\alpha(t)$. To this end, we build two deep neural networks $u_\theta(x,t)$ and $\alpha_\delta(t)$ with θ and δ as independent parameter spaces to approximate the solutions $u(x,t)$ and $\alpha(t)$. Then, we approximate the area temperature solution and the moving boundary by minimizing a composite loss function.

3.3.2. Adding Additional Terms to the Loss Function

The original PINN loss function is shown in (9), and in this paper, we considered adding a loss function with small sample data, as follows:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{f}}\mathcal{L}_{\text{f}} + \lambda_{\text{ic}}\mathcal{L}_{\text{ic}} + \lambda_{\text{bc}}\mathcal{L}_{\text{bc}} + \lambda_{\text{ssl}}\mathcal{L}_{\text{ssl}}, \tag{11}$$

where $\mathcal{L}_{ssl}$ is the loss function for the small sample data, which is used to correct the model. Adding small sample data can improve the prediction accuracy of the model, especially in the case of fewer data. Meanwhile, it can improve the generalization ability and robustness of the model to avoid the risk of overfitting.

3.3.3. Loss Function Improvement Strategy

The mean-squared error (MSE) is the most-commonly used regression loss function in deep learning and is the sum of squares of the distance between the target variable and the predicted value. Considering the specificity of Stefan problem, sharp or irregular regions will appear during the melting or freezing process, which leads to outliers, and the mean-squared error is very sensitive to outliers; thus, we introduce the log-cosh loss function, for

a small error; $\log(\cosh(x))$ is similar to $\frac{x^2}{2}$, while for large errors, $\log(\cosh(x))$ is similar to $|x| - \log 2$, which means that the log-cosh loss function can have the advantage of the mean-squared error while not being affected by too many outliers. It is also quadratically derivable at every point. The introduction of the log-cosh loss function allows for better model generality and model prediction accuracy. The log-cosh loss function is expressed as follows:

$$\mathcal{L}(Y, \hat{Y}) = \sum_{i=1}^{n} \log\left(\cosh\left(\hat{Y}_i - Y_i\right)\right), \tag{12}$$

where the log-cosh loss function is smooth and has a continuous derivative property, which allows the model to be trained using derivative-based optimization algorithms such as gradient descent, and the log-cosh loss function is a symmetric loss function; it has equal loss for positive and negative errors. This symmetry allows the model to better handle both positive and negative errors, thus improving the model fitting ability.

To summarize, our proposed algorithm is displayed in Algorithm 1 and visualized in the schematic diagram in Figure 3, where we explain each step in detail. First, we construct two neural networks $u_\theta(x,t)$ and $\alpha_\delta(t)$. $\theta = \left\{W^k, b^k\right\}_{1 \leqslant k \leqslant K}$ is the set of weight matrices and bias vectors in u; $\delta = \left\{W^j, b^j\right\}_{1 \leqslant j \leqslant J}$ is the set of weight matrices and bias vectors in α; for the neural networks as a substitute for u and α, we can use auto-differentiation and the chain rule to differentiate the functions, then we need to restrict the two neural networks to satisfy the physical conditions imposed by the PDE and Stefan condition, while it is harder to achieve if we constrain over the whole region, so we constrain u and α at scattered points and various training sets and measure the difference between the two neural networks and the constraint; we define the loss function as the sum of the hyperbolic cosine logarithms of the equations and the residuals of the boundary conditions (where the derivatives involved are handled automatically by AD), and in the last step, we minimize the total loss function to obtain the optimal parameters and, thus, the optimal solution. Since the loss is nonlinear and nonconvex, we usually use gradient-based optimizers to minimize the loss function, such as gradient descent, Adam [36], and L-BFGS [37].

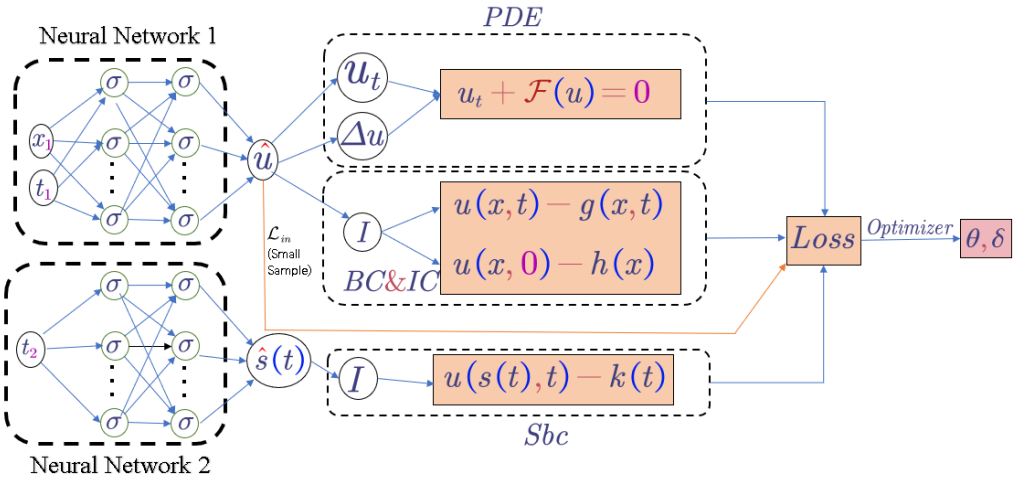

Figure 3. Stefan neural network structure diagram.

Algorithm 1 Deep neural networks for solving the Stefan problem.

1: Two neural networks $u_\theta(x,t)$ and $\alpha_\delta(t)$ with θ and δ as parameters and propagating forward independently of each other are constructed.
2: The training sets $\mathcal{K}_{\mathrm{f}}$ for the temperature control equation, the training sets $\mathcal{K}_{\mathrm{u}_0}$ and $\mathcal{K}_{\mathrm{u}_{\mathrm{nc}}}$ for the initial margin condition, the training sets $\mathcal{K}_{\delta_0}$ for the initial margin condition of the moving interface, $\mathcal{K}_{\delta_{\mathrm{nc}}}$, $\mathcal{K}_{\delta_{\mathrm{ic}}}$, $\mathcal{K}_{\delta_{\mathrm{bc}}}$, and the training set $\mathcal{K}_{\mathrm{u}_{\mathrm{ssl}}}$ for the residuals of the internal small sample data.
3: The total loss function is assigned by summing the hyperbolic cosine logarithms of the control equation, the initial margin conditional residuals, the moving boundary initial margin conditional residuals, and the interior point residuals.
4: The optimal parameters θ and δ are obtained by minimizing the loss function and training the neural network to obtain the optimal solution.

4. Numerical Examples

In this section, we consider several two-dimensional classical forward and inverse Stefan problems to demonstrate the robustness and power of the proposed model. We validated the proposed model for different numbers of small sample data points (0, 50, 100, 200). The results showed that small sample data can effectively improve the detection accuracy. In Section 4.1, one two-dimensional forward Stefan problem is given to test the accuracy of the proposed model, and in Section 4.2, two two-dimensional inverse Stefan problems are given to verify the generality of the model.

For error evaluation, we used the relative L^2 norm:

$$Error = \frac{\sqrt{\sum_{j=1}^{M} |u^j_{\mathrm{exact}} - u^j_{\mathrm{pred}}|^2}}{\sqrt{\sum_{j=1}^{M} |u^i_{\mathrm{exact}}|^2}}. \tag{13}$$

Here, M represents the number of all points in the training process of two mutually independent neural networks, u_{pred} represents the predicted values of the corresponding coordinate points, and u_{exact} represents the exact values of the corresponding coordinate points.

4.1. Two-Dimensional Single-Phase Stefan Forward Problem

In this subsection, a classical two-dimensional solid melting problem model is used as an example to illustrate the effectiveness of the method [38].

As depicted in Figure 4, let $D \in \mathbb{R}^2$, for any $0 < t < T$, $\Omega(t)$, be the time-dependent bounded subdomain in D, $\partial\Omega(t)$ be its boundary, and $\partial\Omega(t) = \Gamma(t) \cup \Sigma(t)$, where $\Gamma = \cup_{0<t<T}\Gamma(t)$ represents the fixed boundary and $\Sigma = \cup_{0<t<T}\Sigma(t)$ represents the moving boundary.

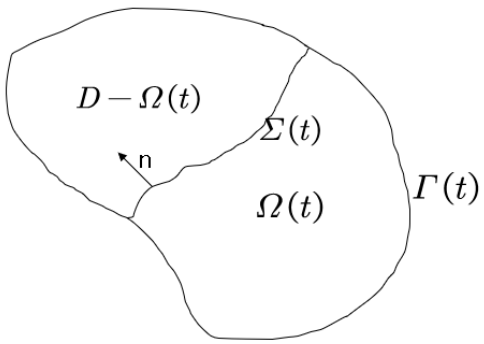

Figure 4. Stefan neural networks structure diagram.

The above Stefan problem can be described in mathematics as

$$u_t - \Delta u = 0 \ in\Omega, \tag{14}$$
$$u|_{\overline{\Gamma}} = g, \tag{15}$$
$$u|_{\Omega(0)} = u_0, \tag{16}$$
$$\Sigma(0) = \Sigma_0, \tag{17}$$
$$u|_{\overline{\Sigma}} = u^*, \tag{18}$$
$$\frac{\partial u}{\partial n}|_{\overline{\Sigma}} = h, \tag{19}$$

where Δ is the Laplace operator for spatial variables, n is the normal vector of the definition field Ω with respect to the time variables, and the definition field Ω is

$$\Omega(t) = \left\{(x, y, t) \in \mathbb{R}^3 : 0 < x < \alpha(y, t), 0 < y < 1\right\} \subset \Omega^*. \tag{20}$$

$\alpha(y, t)$ denotes the unknown free boundary; the artificial region is defined as: $\Omega^* = [0, 2.25] \times [0, 1] \times [0, 1]$, defining $\Sigma(t)$ for

$$\Sigma(t) = \left\{(x, y, t) \in \mathbb{R}^3 : \Phi(x, y, t) = 0, 0 < y < 1, 0 \leqslant t \leqslant 1\right\}, \tag{21}$$

where $\Phi(x, y, t) = x - \alpha(y, t)$. Define the following parameters:

$$\begin{aligned} &u_0(x, y) = \exp\left(-x + \frac{1}{2}y + \frac{1}{2}\right), \quad x, y \in \Omega(0), \\ &u(\alpha(y, t), y, t) = u^* = 0, \\ &h = \frac{1}{|\nabla\Phi|}\frac{\partial\Phi}{\partial t}, \ \alpha(y, 0) = \alpha_0(y) = \frac{1}{2}y + \frac{1}{2}. \end{aligned} \tag{22}$$

The boundary conditions are defined as

$$\begin{aligned} &u(x, 0, t) = g_1(x, t) = \exp\left(1.25t - x + \frac{1}{2}\right) - 1, \ (x, 0, t) \in \Omega, \\ &u(0, y, t) = g_2(x, t) = \left(1.25t + 0.5y + \frac{1}{2}\right) - 1, \ (0, y, t) \in \Omega, \\ &u(x, 1, t) = g_3(x, t) = \exp(1.25t - x + 1), \ (x, 1, t) \in \Omega. \end{aligned} \tag{23}$$

Construct the following true solution:

$$\begin{aligned} &u(x, y, t) = \exp\left(\frac{5}{4}t - x + \frac{1}{2}y + \frac{1}{2}\right) - 1, \\ &\alpha(y, t) = \frac{1}{2}y + \frac{5}{4}t + \frac{1}{2}. \end{aligned} \tag{24}$$

As shown in Algorithm 1, we built two fully connected neural networks $u_\theta(x, y, t)$ and $\alpha_\delta(y, t)$ that propagate independently of each other to approximate the solution of the control equation $u(x, y, t)$ and the moving boundary $\alpha(y, t)$, and we trained these two neural networks by minimizing the loss function.

$$\mathcal{L}(\theta, \delta) = \mathcal{L}_{\mathrm{f}}(\theta) + \mathcal{L}_{\mathrm{u_0}}(\theta) + \mathcal{L}_{\mathrm{u_{bc}}}(\theta) + \mathcal{L}_{\alpha_0}(\delta) + \mathcal{L}_{\alpha_{\mathrm{bc}}}(\theta, \delta) + \mathcal{L}_{\alpha_{\mathrm{nc}}}(\theta, \delta) + \mathcal{L}_{u_{\mathrm{ssl}}}(\theta), \tag{25}$$

where

$$\mathcal{L}_{\mathrm{f}}(\theta) = \sum_{i=1}^{N_f} \log\left(\cosh\left(u_\theta\left(x_{\mathrm{f}}^i, y_{\mathrm{f}}^i, t_{\mathrm{f}}^i\right) - u\left(x_{\mathrm{f}}^i, y_{\mathrm{f}}^i, t_{\mathrm{f}}^i\right)\right)\right),$$

$$\mathcal{L}_{u_0}(\theta) = \sum_{j=1}^{N_{u_0}} \log\left(\cosh\left(u_\theta\left(x_{u_0}^j, y_{u_0}^j, 0\right) - u_0\left(x_{u_0}^j, y_{u_0}^j, 0\right)\right)\right),$$

$$\begin{aligned}\mathcal{L}_{u_{bc}}(\theta) = &\sum_{k=1}^{N_{u_{bc}}} \log\left(\cosh\left(u_\theta\left(x_{u_{bc}}^k, 0, t_{u_{bc}}^k\right) - g_1\left(x_{u_{bc}}^k, t_{u_{bc}}^k\right)\right)\right)\\ &+ \sum_{k=1}^{N_{u_{bc}}} \log\left(\cosh\left(u_\theta\left(0, y_{u_{bc}}^k, t_{u_{bc}}^k\right) - g_2\left(y_{u_{bc}}^k, t_{u_{bc}}^k\right)\right)\right)\\ &+ \sum_{k=1}^{N_{u_{bc}}} \log\left(\cosh\left(u_\theta\left(x_{u_{bc}}^k, 1, t_{u_{bc}}^k\right) - g_2\left(x_{u_{bc}}^k, t_{u_{bc}}^k\right)\right)\right),\end{aligned}$$

$$\mathcal{L}_{\alpha_0}(\delta) = \sum_{j=1}^{N_{\alpha_0}} \log\left(\cosh\left(\alpha_\delta\left(y_{\alpha_0}^j, 0\right) - \alpha_0\left(y_{\alpha_0}^j\right)\right)\right),$$

$$\mathcal{L}_{\alpha_{bc}}(\theta, \delta) = \sum_{k=1}^{N_{\alpha_{bc}}} \log\left(\cosh\left(u_\theta\left(\alpha_\delta\left(y_{bc}^k, t_{bc}^k\right), y_{bc}^k, t_{bc}^k\right)\right)\right),$$

$$\mathcal{L}_{\alpha_{nc}}(\theta, \delta) = \sum_{k=1}^{N_{\alpha_{nc}}} \log\left(\cosh\left(\frac{\partial u_\theta}{\partial n}\left(\alpha_\delta\left(y_{nc}^k, t_{nc}^k\right), y_{nc}^k, t_{nc}^k\right) - h\left(y_{nc}^k, t_{nc}^k\right)\right)\right),$$

$$\mathcal{L}_{u_{ssl}}(\theta) = \sum_{j=1}^{N_{u_{ssl}}} \log\left(\cosh\left(u_\theta\left(x_{ssl}^j, y_{ssl}^j, t_{ssl}^j\right) - u\left(x_{ssl}^j, y_{ssl}^j, t_{ssl}^j\right)\right)\right).$$

In particular, since the moving interface $\alpha(y,t)$ is unknown in advance, the residual point region of the equation was set to the artificial region Ω^*, and the neural networks was trained to restrict the neural network prediction solution to the defined domain Ω. Two Stefan neural networks $u_\theta(x,t)$ and $\alpha_\delta(t)$ were constructed with independent forward propagation, and the neural network parameters were set as follows: $u_\theta(x,t)$ has a network structure of $3 + 100 \times 3 + 1$; the input layer has three variables and three hidden layers; each layer contains 100 neurons; the output layer outputs the neural networks prediction solution. The network structure of $\alpha_\delta(t)$ is $2 + 100 \times 3 + 1$; similarly, the input layer has two variables, and the output layer has three hidden layers containing 100 neurons each, while the moving boundary of the neural network prediction is the output. The training set was selected as N_f with 256 points, 256 points for each of the initial border conditions (including the initial border conditions on the free boundary), and 0, 50, 100, and 200 for the small sample data points in the region. The results are shown in Table 1. The same initial learning rate of 10^{-3} was set for both neural networks, and the number of iterations was 40,000 with Adam's algorithm, using tanh as the activation function and using Xavier to initialize both neural networks.

Table 1. Effect of different small sample data points on two-dimensional forward Stefan problem.

Small Sample Data	0 (Original [34])	50	100	150	200
Solutions' L^2 error	8.12×10^{-2}	1.12×10^{-2}	8.57×10^{-3}	9.87×10^{-3}	3.32×10^{-2}
Boundary L^2 error	4.32×10^{-2}	4.22×10^{-2}	3.25×10^{-2}	4.33×10^{-2}	4.45×10^{-2}

By training the neural networks with the above parameters, Figure 5 shows the L^2 error plot of the predicted solution and the true solution, and Figure 6 shows the L^2 error plot of the predicted free interface $\alpha(y,t)$ and the true free boundary of the neural networks. It is worth mentioning that the number of small samples u_{ssl} is set to 100 in Figures 5 and 6, and we can see that the L^2 errors reached 8.57×10^{-3} and 3.25×10^{-2}, respectively, which is a significant improvement in accuracy compared with the original neural networks [34].

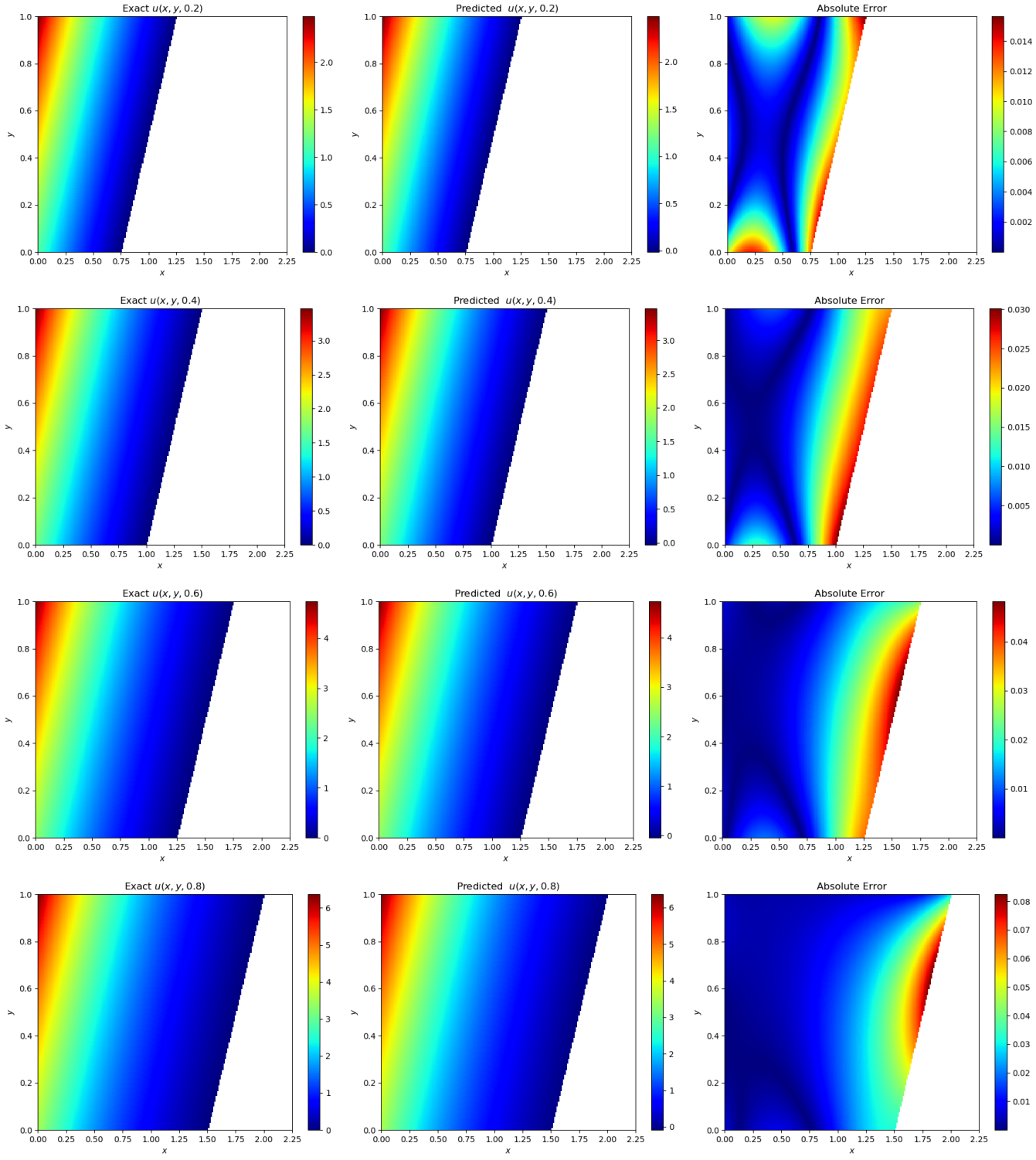

Figure 5. Improved physics-informed neural networks for solving the two-dimensional forward Stefan problem solution error diagram.

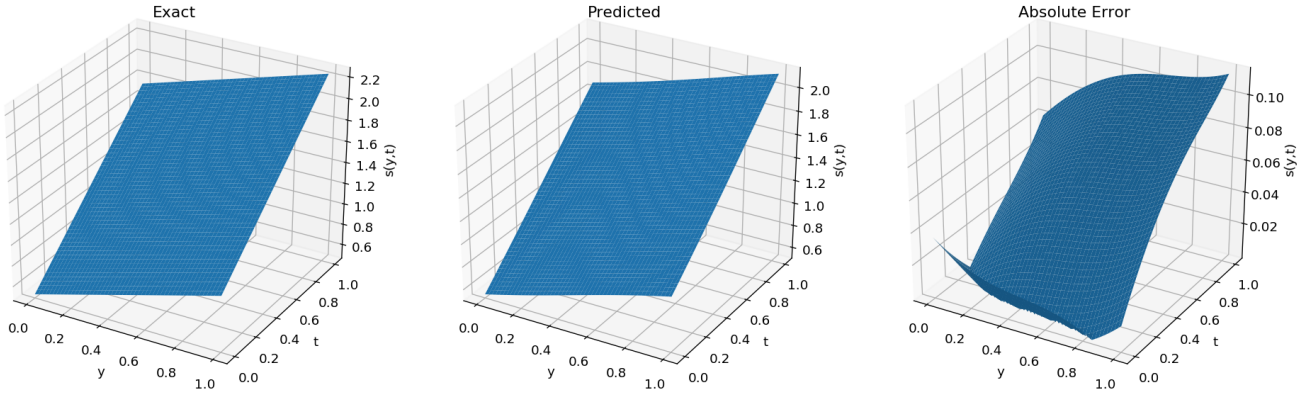

Figure 6. Improved physics-informed neural networks for solving the two-dimensional forward Stefan problem to predict the free boundary error diagram.

4.2. Two-Dimensional Single-Phase Stefan Inverse Problem I

For the two-dimensional single-phase Stefan inverse problem, we assumed that the boundary conditions are not known and provide additional information at the final moment T:

$$\mathrm{u}|_{\Omega(\mathrm{T})} = \mathrm{u}_{\mathrm{T}}. \tag{26}$$

Then, this class of inverse problems means that we know the data at the moment T = 1 and need to find the temperature solution $\mathrm{u}(x, y, t)$ and the free boundary $\alpha(y, t)$ and satisfy (14), (16), and (17)–(19).

$$\mathrm{u}(x, y, \mathrm{T}) = \mathrm{u}_{\mathrm{T}} = \exp(1.25 - x + 0.5y + 0.5) - 1. \tag{27}$$

As shown in Algorithm 1, we built two fully connected neural networks $u_\theta(x, y, t)$ and $\alpha_\delta(y, t)$ that propagate independently of each other to approximate the solution of the control equation $u(x, y, t)$ and the moving boundary $\alpha(y, t)$, and we trained these two neural networks by minimizing the loss function.

$$\mathcal{L}(\theta, \delta) = \mathcal{L}_{\mathrm{r}}(\theta) + \mathcal{L}_{\mathrm{u}_0}(\theta) + \mathcal{L}_{\mathrm{u}_{\mathrm{T}}}(\theta) + \mathcal{L}_{\alpha_{\mathrm{bc}}}(\theta, \delta) + \mathcal{L}_{\alpha_{\mathrm{nc}}}(\theta, \delta) + \mathcal{L}_{\alpha_0}(\delta) + \mathcal{L}_{\mathrm{u}_{\mathrm{ssl}}}(\theta), \tag{28}$$

where the loss function definitions are all the same as in Section 4.1, except that $\mathcal{L}_{\mathrm{u}_{\mathrm{T}}}(\theta)$:

$$\mathcal{L}_{\mathrm{u}_{\mathrm{T}}}(\theta) = \sum_{\mathrm{j}=1}^{\mathrm{m}} \log\left(\cosh\left(\mathrm{u}_\theta\left(x^{\mathrm{j}}, y^{\mathrm{j}}, 1\right) - \mathrm{u}\left(x^{\mathrm{j}}, y^{\mathrm{j}}, 1\right)\right)\right). \tag{29}$$

We trained the neural networks using exactly the same parameters as in Section 4.1, as shown in Table 2, which visualizes the L^2 error of the solution and moving boundaries when the small sample data are 0, 50, 100, 150, and 200, respectively.

Table 2. Effect of different small sample data points on the two-dimensional inverse Stefan problem I.

Small Sample Data	0 (Original [34])	50	100	150	200
Solutions' L^2 error	8.89×10^{-2}	3.82×10^{-2}	3.66×10^{-2}	4.49×10^{-2}	4.59×10^{-2}
Boundary L^2 error	3.41×10^{-2}	3.35×10^{-2}	3.15×10^{-2}	3.32×10^{-2}	3.36×10^{-2}

Figure 7 gives the error plots of the predicted and exact temperature solutions at moments t = 0.2, 0.4, 0.6, and 0.8, and Figure 8 shows the error plots of the predicted moving boundaries and the true moving boundaries. In Figures 7 and 8, the number of small samples u_{ssl} was set to 100, and we can see that the L^2 error reached 3.66×10^{-2} and 3.15×10^{-2}, which improved the accuracy of the neural networks compared with the original neural networks.

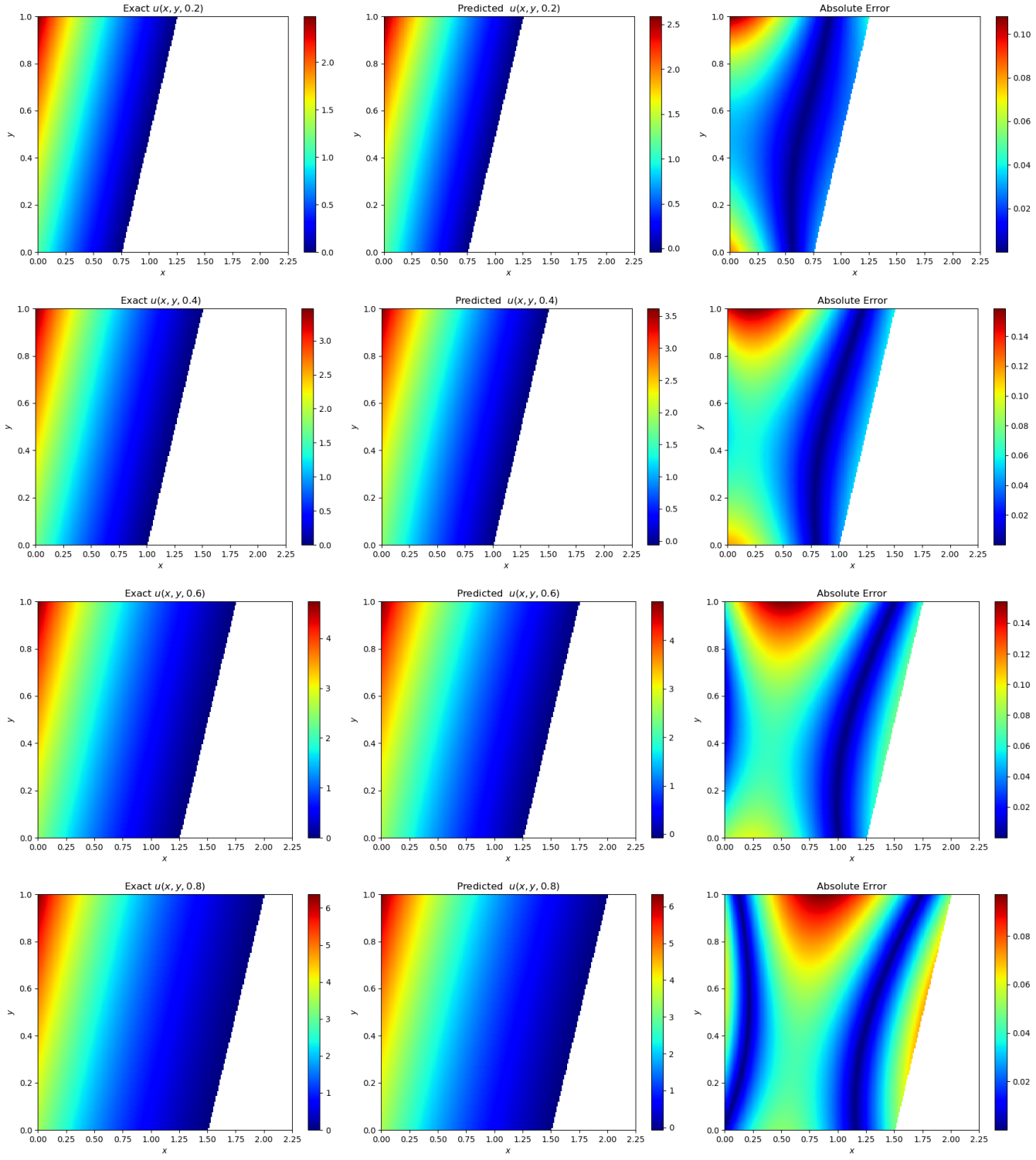

Figure 7. Improved physics-informed neural networks for solving the two-dimensional inverse Stefan problem solution error diagram.

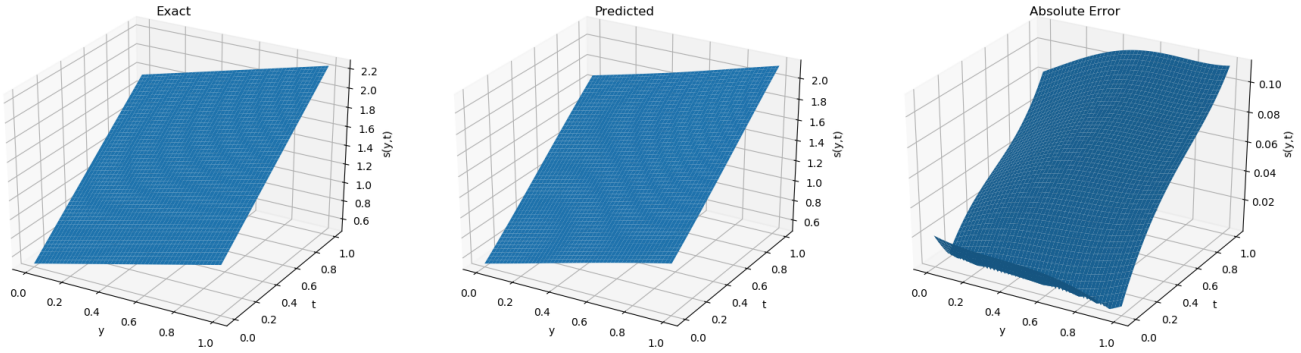

Figure 8. Improved physics-informed neural networks for solving the two-dimensional inverse Stefan problem I to predict the free boundary error diagram.

4.3. Two-Dimensional Single-Phase Stefan Inverse Problem II

This part mainly discusses the Stefan problem of inversion of the phase change boundary in the case of known temperature and heat changes at the boundary of a homogeneous medium, without considering any initial conditions or boundary conditions at the moving interface; only some given temperature measurements in the definition domain were considered, and neural networks were used to approximate the temperature solution and the unknown position of the moving boundary, then the above inverse problem is described by the mathematical formula:

$$\begin{aligned} &u_t - \Delta u = 0,\ (x,y) \in \Omega, \\ &\Sigma(0) = \Sigma_0, \\ &u|_{\overline{\Sigma}} = u^*, \\ &\frac{\partial u}{\partial n}|_{\overline{\Sigma}} = h. \end{aligned} \tag{30}$$

Similarly, two fully connected neural networks $u_\theta(x,y,t)$ and $\alpha_\delta(y,t)$ that propagate independently of each other were constructed to approximate the solution of the equation $u(x,y,t)$ and the moving boundary $\alpha(y,t)$, and we trained these two neural networks by minimizing the following loss functions:

$$\mathcal{L}(\theta,\delta) = \mathcal{L}_{\text{data}}(\theta) + \mathcal{L}_{\text{f}}(\theta) + \mathcal{L}_{\alpha_{\text{bc}}}(\theta,\delta) + \mathcal{L}_{\alpha_{\text{nc}}}(\theta,\delta) + \mathcal{L}_{\text{u}_{\text{ssl}}}(\theta), \tag{31}$$

where

$$\mathcal{L}_{data}(\theta) = \sum_{i=1}^{N_{data}} \log\left(\cosh\left(u_\theta\left(x^i_{data}, y^i_{data}, t^i_{data}\right) - u^i\left(x^i, y^i, t^i\right)\right)\right),$$

$$\mathcal{L}_{\text{f}}(\theta) = \sum_{i=1}^{N_{\text{f}}} \log\left(\cosh\left(u_\theta\left(x^i_{\text{f}}, y^i_{\text{f}}, t^i_{\text{f}}\right) - u\left(x^i_{\text{f}}, y^i_{\text{f}}, t^i_{\text{f}}\right)\right)\right),$$

$$\mathcal{L}_{\alpha_{\text{bc}}}(\theta,\delta) = \sum_{k=1}^{N_{\alpha_{\text{bc}}}} \log\left(\cosh\left(u_\theta\left(\alpha_\delta\left(y^k_{\text{bc}}, t^k_{\text{bc}}\right), y^k_{\text{bc}}, t^k_{\text{bc}}\right)\right)\right),$$

$$\mathcal{L}_{\alpha_{\text{nc}}}(\theta,\delta) = \sum_{k=1}^{N_{\alpha_{\text{nc}}}} \log\left(\cosh\left(\frac{\partial u_\theta}{\partial n}\left(\alpha_\delta\left(y^k_{\text{nc}}, t^k_{nc}\right), y^k_{\text{nc}}, t^k_{\text{nc}}\right) - h\left(y^k_{\text{nc}}, t^k_{\text{nc}}\right)\right)\right),$$

$$\mathcal{L}_{u_{ssl}}(\theta) = \sum_{j=1}^{N_{u_{ssl}}} \log\left(\cosh\left(u_\theta\left(x_{ssl}^j, y_{ssl}^j, t_{ssl}^j\right) - u\left(x_{ssl}^j, y_{ssl}^j, t_{ssl}^j\right)\right)\right).$$

In particular, the first loss function was constructed from the residuals of the temperature observation points in the region, where $\{(x^i, y^i, t^i), u^i\}_{i=1}^N$ is a randomly sampled data point in the domain. The neural networks parameters were set as follows: the network structures of the two neural networks $u_\theta(x,t)$ and $\alpha_\delta(t)$ were the same as in the previous example, and the training set was selected as follows: the residuals of the equations and the initial margin conditions (including the initial margin conditions on the free boundary) were each selected as 256 points. The results are shown in Table 3. The initial learning rate was set to 10^{-3} for both neural networks, and the number of iterations was set to 40,000 and optimized by Adam optimizer. Xavier initialized both neural networks.

Table 3. Effect of different small sample data points on the two-dimensional inverse Stefan problem II.

Small Sample Data	0 (Original [34])	50	100	150	200
Solutions' L^2 error	6.12×10^{-2}	5.12×10^{-2}	9.97×10^{-3}	4.32×10^{-2}	7.62×10^{-2}
Boundary L^2 error	3.04×10^{-2}	2.13×10^{-2}	1.26×10^{-2}	2.56×10^{-2}	4.25×10^{-2}

From Figures 9 and 10, it can be seen that the L^2 error of the solution of the inverse problem and the moving interface can reach 9.97×10^{-3} and 1.26×10^{-2} for the improved deep neural networks with small sample data of 100, which is a significant improvement compared with the original neural network.

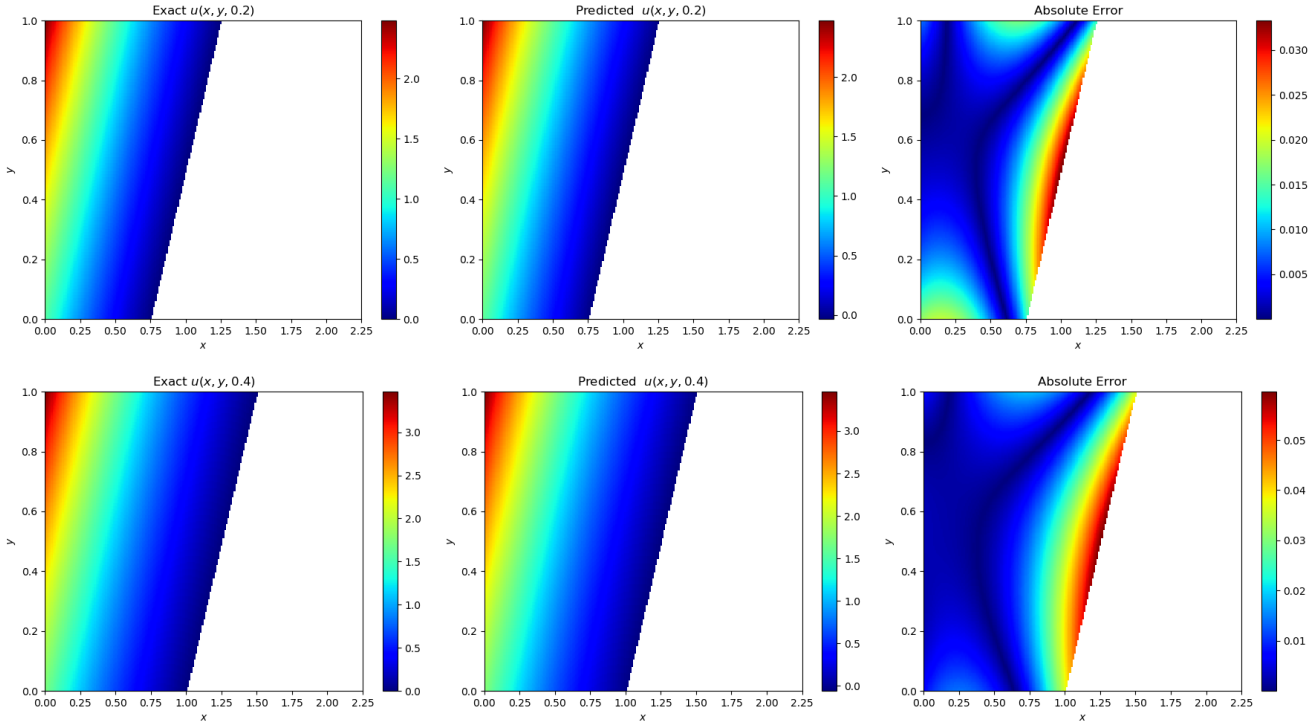

Figure 9. *Cont.*

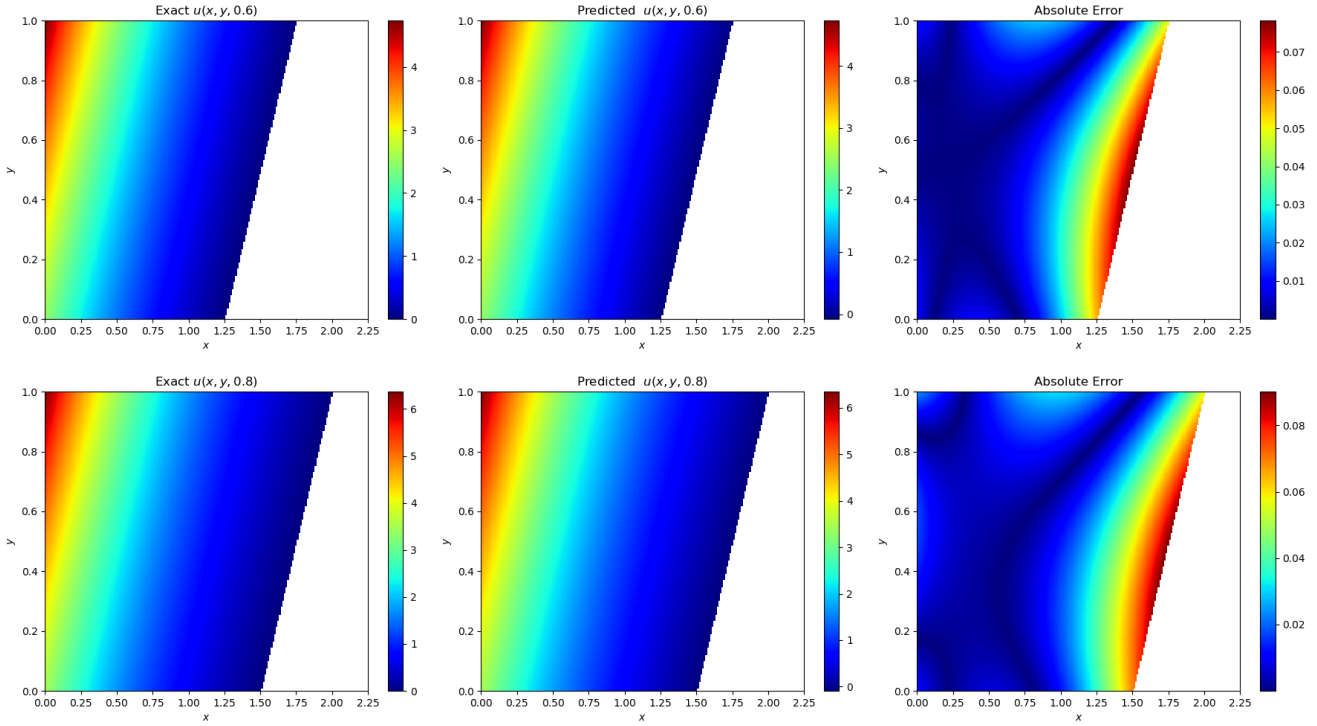

Figure 9. Improved physics-informed neural networks for solving the two-dimensional inverse Stefan problem solution error diagram.

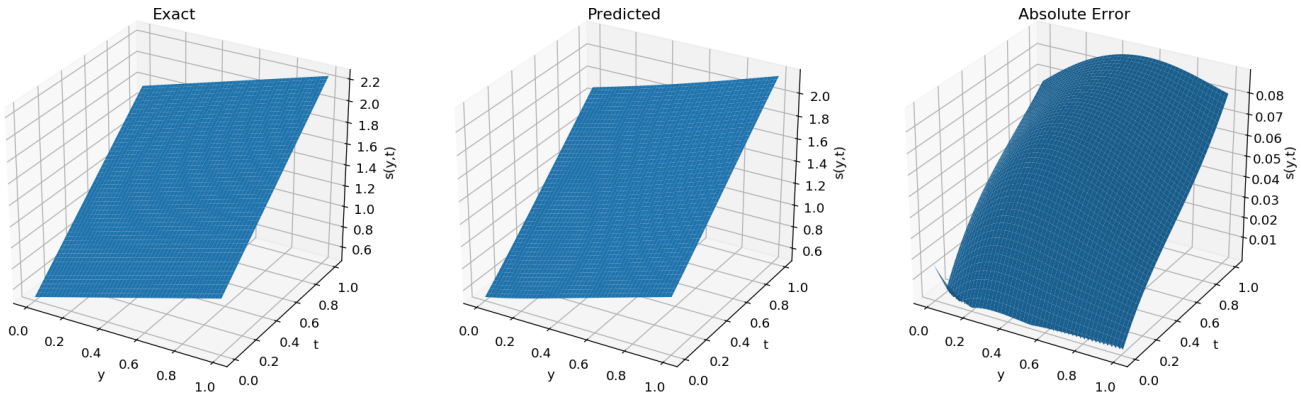

Figure 10. Improved physics-informed neural networks for solving the two-dimensional inverse Stefan problem II to predict the free boundary error diagram.

The above two-dimensional single-phase Stefan inverse problem II is based on the temperature measurement value $M = 50$, and then, we tested the performance of our proposed neural network framework by adding white noise κ with magnitude equal to 1%, 2%, 5%, and 10% of the L^∞ norm of the solution function $u(x, y, t)$, where the small sample data points were 100. The relative L^2 errors obtained for the prediction solutions $u(x, y, t)$ and $\alpha(y, t)$ are shown in Tables 4 and 5, respectively. We can observe that the prediction accuracy of both $u(x, y, t)$ and $\alpha(y, t)$ improved as the total number of temperature measurement data M increased, but became lower as the noise level κ increased. The latter case verified our conjecture that the dataset became increasingly inaccurate due to higher levels of noise. Another important observation is that, given a sufficient amount of temperature measurement data (e.g., $M = 200$), even with noise corruption up to 10%, the relative L^2 error between the predicted solutions $u(x, y, t)$ and $\alpha(y, t)$ can reach 6.78×10^{-2} and 3.77×10^{-2}, and it can be seen that the improved physics-

informed neural networks with small sample learning can accurately identify the moving boundaries despite the presence of a large amount of noisy data.

Figure 11 shows the drop plots of the loss function for the three cases when the noise level κ is 10%, and it can be seen that the loss function image became relatively stable with the addition of the small sample loss function, which also validated the relevant narratives in Sections 3.3.2 and 3.3.3.

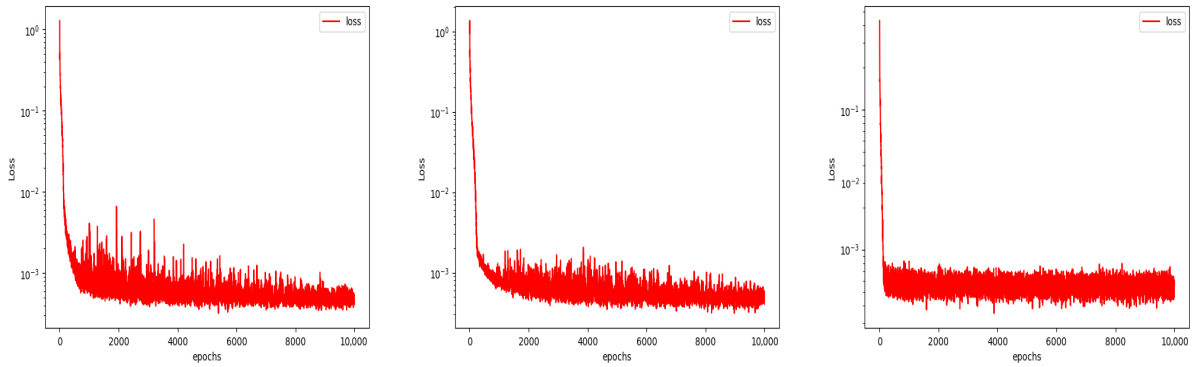

Figure 11. The two-dimensional inverse Stefan problem II with a 10% noise level of the MSE, log-cosh and log-cosh loss function with the addition of small sample loss decreases the plot.

Table 4. Inverse problem II relative L^2 error of the prediction solution $u(x,t)$ for different measurement values M and different noise levels κ.

M \ κ	$\kappa = 0\%$	$\kappa = 1\%$	$\kappa = 2\%$	$\kappa = 5\%$	$\kappa = 10\%$
$M = 50$	9.97×10^{-3}	8.75×10^{-2}	9.87×10^{-2}	1.95×10^{-1}	5.23×10^{-1}
$M = 100$	2.64×10^{-2}	8.87×10^{-2}	9.98×10^{-2}	1.51×10^{-1}	8.36×10^{-1}
$M = 200$	1.85×10^{-2}	2.26×10^{-2}	4.21×10^{-2}	5.75×10^{-2}	6.78×10^{-2}

Table 5. Inverse problem II relative L^2 error of the free boundary $\alpha(t)$ for different measurement values M and different noise levels κ.

M \ κ	$\kappa = 0\%$	$\kappa = 1\%$	$\kappa = 2\%$	$\kappa = 5\%$	$\kappa = 10\%$
$M = 50$	1.26×10^{-2}	1.43×10^{-2}	4.28×10^{-2}	1.72×10^{-1}	1.9×10^{-1}
$M = 100$	1.48×10^{-2}	3.87×10^{-2}	5.11×10^{-2}	8.41×10^{-2}	1.21×10^{-1}
$M = 200$	1.23×10^{-2}	1.35×10^{-2}	2.72×10^{-2}	2.81×10^{-2}	3.77×10^{-2}

4.4. Irregular Area Stefan Problem

To make our proposed algorithmic framework more convincing, the main work of this subsection is to consider numerical solutions and the free boundary of the two-dimensional Stefan problem on an L-shaped complex computational domain that differs from the traditional $[a,b] \times [c,d]$ rectangular domain, which is more capable of testing the performance of our algorithmic framework. It is worth mentioning that the equations and Stefan conditions used in this section are the same as in Section 4.1, except that the boundary information is different due to the complex computational domain.

Figure 12 represents the plan view of the selected training points on the L-shaped complex computational domain. Unlike the above example, we randomly selected the training points on the boundary, and after the complex computational domain was triangulated and dissected, the internal points were selected at the nodes instead of selecting random samples as internal information.

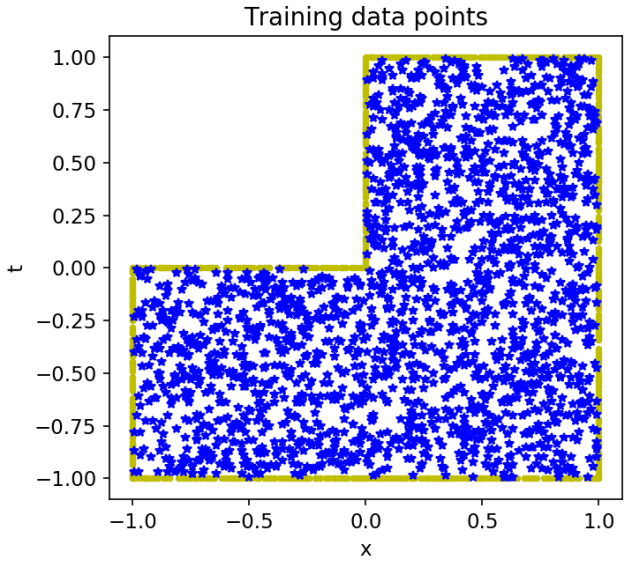

Figure 12. Red circled points indicate boundary points and initial points. Blue star-shaped points indicate internal points.

We used the same hyperparameters in the numerical example above and constructed the same loss function in (15), minimizing the loss function by the Adam optimization algorithm to obtain the optimal solution and the free boundary, as shown in Figure 13, where we can see that our proposed method can also achieve good accuracy for complex computational domains. The relative L^2 error between the predicted solutions $u(x, y, t)$ and $\alpha(y, t)$ can reach 5.74×10^{-3} and 1.56×10^{-2}. Snapshots of moments for $t = 0.2$ s, $t = 0.4$ s, and $t = 0.8$ s are given. It can be seen that the model predicts the temperature solution and moving boundary, in good agreement with the exact solution.

4.5. Irregular Free Boundary Stefan Problem

In this subsection, consider an irregular free boundary problem, the initial shape of which is described by a unit circle of $x^2 + y^2 = 1$ with equations and boundary conditions as described in Section 4.1, $t \in [0, 1]$, and consider the following analytic solution:

$$u(x, y, t) = e^t \Big((t + 1)x^2 + (5t + 1)y^2 - t^2 - 1 \Big) \tag{32}$$

$$\alpha(x, y, t) = (t + 1)x^2 + (5t + 1)y^2 - t^2 - 1 \tag{33}$$

Similar to the description in the previous section, we randomly selected training points on the boundary, and after the complex computational domain was triangulated and dissected, internal points were selected at the nodes. The distribution of training points is shown in the Figure 14.

Next, we also used the same loss function construction as in (15) to predict $u(x, y, t)$ and $\alpha(x, y, t)$. We give snapshots of the predictions when $t = 0.2$ s, $t = 0.4$ s, and $t = 0.8$ s in Figure 15. From that, we can see that our proposed neural network framework can also have a good prediction for general irregular moving boundaries; meanwhile, the relative L^2 error between the predicted solutions $u(x, y, t)$ and $\alpha(x, y, t)$ can reach 3.51×10^{-2} and 2.47×10^{-2}.

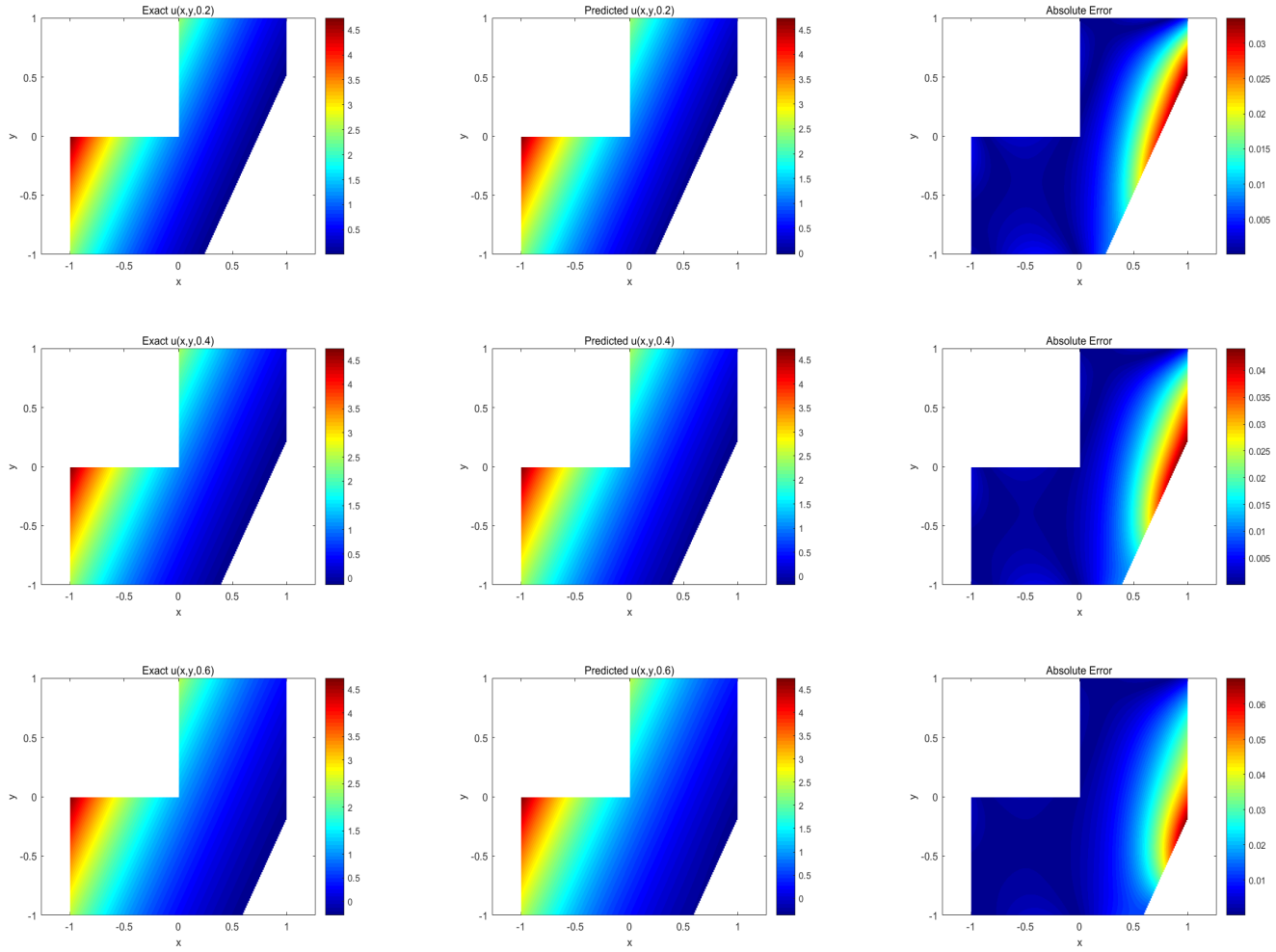

Figure 13. Exact solution, prediction solution, point-by-point error, and free boundary location for different moments of the temperature in the irregular region of the Stefan problem.

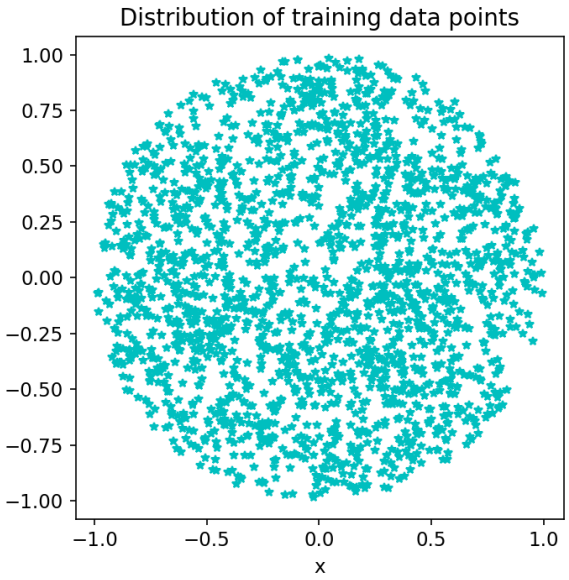

Figure 14. Scatter plot of training points.

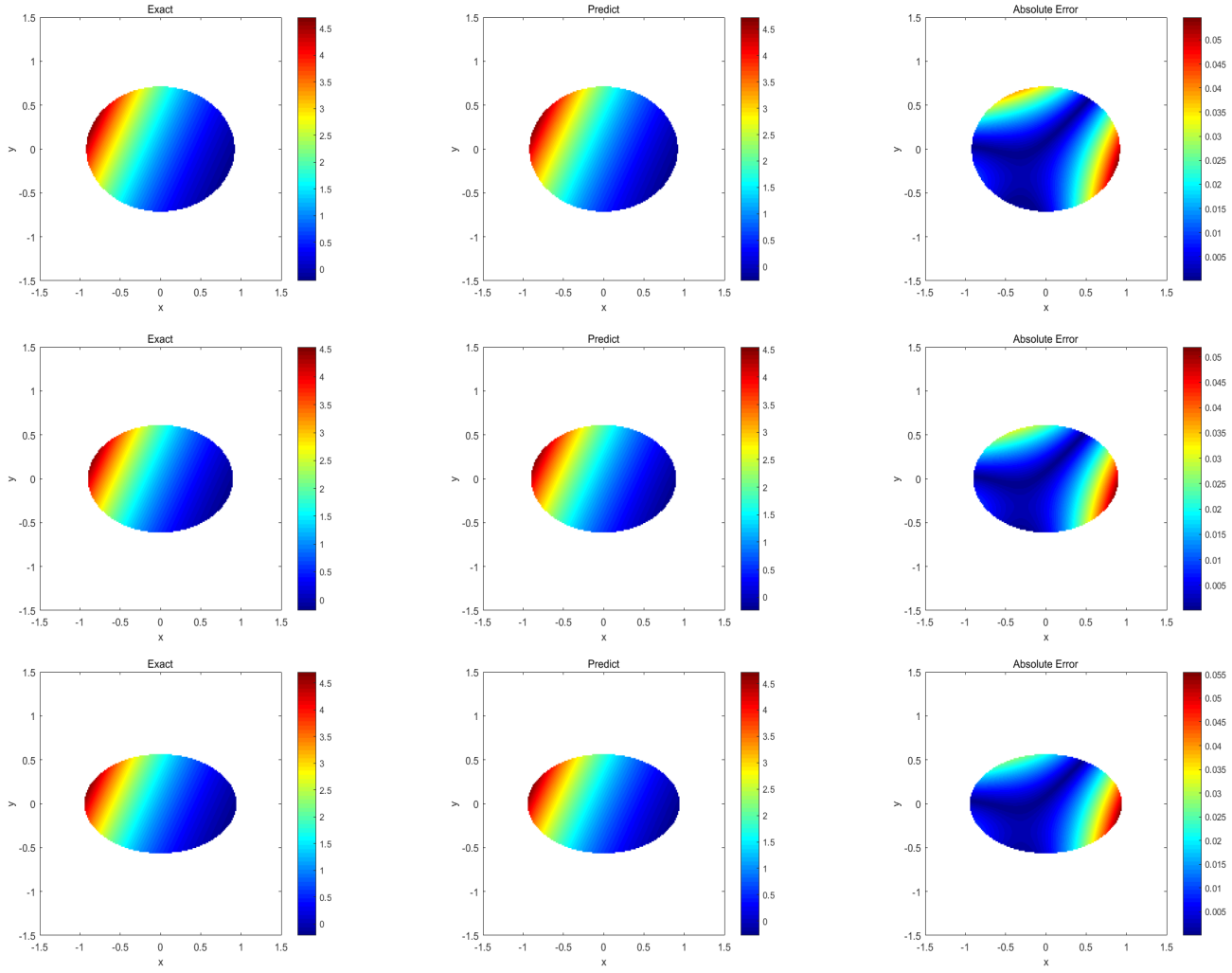

Figure 15. Exact solution, prediction solution, point-by-point error, and free boundary location for different moments of temperature in the irregular boundary of the Stefan problem.

5. Conclusions

In this paper, by improving the neural network structure of physics-informed neural networks (PINNs) and combining the idea of small sample learning, a generalized neural network framework for solving the Stefan problems was proposed, which can be directly applied to various types of Stefan problems with only small changes. After several numerical experiments, it was proven that the improved deep neural networks based on small sample learning improved the computational accuracy by 2–3-times compared with the original neural network, and the model generalization ability was significantly improved. Meanwhile, this paper demonstrated that the proposed method had a good prediction effect and accuracy through the examples of the irregular region and irregular free boundary. The study of Stefan problems with sharp, irregular geometries and topological variations, as well as three-dimensional or even higher-dimensional Stefan problems is the focus of the future work.

Author Contributions: Conceptualization, X.F.; methodology, J.L. and W.W.; software, J.L.; validation, X.F.; formal analysis, X.F. and J.L.; investigation, J.L.; resources, X.F.; data curation, J.L.; writing—original draft preparation, J.L.; writing—review and editing, X.F. and J.L.; visualization, J.L.; supervision, X.F.; project administration, X.F.; funding acquisition, X.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work is in parts supported by the National Natural Science Foundation of Xinjiang Province (No. 2022TSYCTD0019, No. 2022D01D32, No. 2020D04002).

Acknowledgments: The authors would especially like to thank the editors and reviewers for their suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Crank, J. *Free and Moving Boundary Problems*; Springer: London, UK, 1984.
2. Flemings, M.C. *Solidification Processing*; McGraw-Hill: New York, NY, USA, 1984.
3. Canic, S.; Keyfit, B.L.; Lieberma, G.M.; Wang, X. A proof of existence of perturbed steady transonic shocks via a free boundary problem. *Comm. Pure Appl. Math.* **2000**, *53*, 484–511. [CrossRef]
4. Berestycki, H.; Brezis, H. On a free boundary problem arising in plasma physics. *Nonlinear Anal.* **1980**, *4*, 415–436. [CrossRef]
5. Sandier, E.; Serfaty, S. A rigorous derivation of a free-boundary problem arising in superconductivity. *Ann. Sci. l'École Norm. Super.* **2000**, *33*, 561–592. [CrossRef]
6. Kolodner, I.I. Free boundary problem for the heat equation with applications to problems of change of phase. I. General method of solution. *Comm. Pure Appl. Math.* **1956**, *9*, 1–31. [CrossRef]
7. Friedman, A. Free boundary problems in science and technology. *Not. Am. Math. Soc.* **2000**, *47*, 854–861.
8. Merz, W.; Rybka, P. A free boundary problem describing reaction-diffusion problems in chemical vapor infiltration of pyrolytic carbon. *J. Math. Anal. Appl.* **2004**, *292*, 571–588. [CrossRef]
9. Chen, X.F.; Friedman, A. A free boundary problem for an elliptic-hyperbolic system: An application to tumor growth. *SIAM J. Math. Anal.* **2003**, *35*, 974–986. [CrossRef]
10. Cui, S.B. Well-posedness of a multidimensional free boundary problem modelling the growth of nonnecrotic tumors. *J. Funct. Anal.* **2007**, *245*, 1–18. [CrossRef]
11. Friedman, A.; Hu, B. Bifurcation for a free boundary problem modeling tumor growth by Stokes equation. *SIAM J. Math. Anal.* **2007**, *39*, 174–194. [CrossRef]
12. Tao, Y. A free boundary problem modeling the cell cycle and cell movement in multicellular tumor spheroids. *J. Differ. Equ.* **2009**, *247*, 49–68. [CrossRef]
13. Xu, S. Analysis of a delayed free boundary problem for tumor growth. *Discret. Contin. Dyn. Syst. Ser. B.* **2011**, *15*, 293–308. [CrossRef]
14. Du, Y.; Guo, Z.M. Spreading-vanishing dichotomy in a diffusive logistic model with a free boundary. *J. Differ. Equ.* **2011**, *250*, 4336–4366. [CrossRef]
15. Du, Y.; Lin, Z.G. Spreading-vanishing dichotomy in the diffusive logistic model with a free boundary. *SIAM J. Math. Anal.* **2010**, *42*, 377–405. [CrossRef]
16. Lin, Z.G. A free boundary problem for a predator-prey model. *Nonlinearity* **2007**, *20*, 1883–1892. [CrossRef]
17. Du, Y.; Lou, B. Spreading and vanishing in nonlinear diffusion problem with free boundaries. *Anal. PDEs* **2013**, *1301*, 5737. [CrossRef]
18. Goodman, J.; Ostrov, D.N. On the early exercise boundary of the American put option. *SIAM J. Appl. Math.* **2002**, *62*, 1823–1835.
19. Liang, J.; Hu, B.; Jiang, L. Optimal convergence rate of the binomial tree scheme for American options with jump diffusion and their free boundaries. *SIAM J. Financ. Math.* **2010**, *1*, 30–65. [CrossRef]
20. Caldwell, J.; Kwan, Y.Y. Numerical methods for one-dimensional Stefan problems. *Commun. Numer. Meth. Engng.* **2004**, *20*, 535–545. [CrossRef]
21. Juric, D.; Tryggvason, G. A front tracking method for dendritic solidification. *J. Comput. Phys.* **1996**, *123*, 127–148. [CrossRef]
22. Segal, G.; Vuik, C.; Vermolen, F. A conserving discretization for the free boundary in a two-dimensional Stefan problem. *J. Comput. Phys.* **1998**, *141*, 1–21. [CrossRef]
23. Murray, W.D.; Landis, F. Numerical and machine solutions of transient heat-conduction problems involving melting or freezing. *Trans. ASME J.* **1959**, *81*, 106–112. [CrossRef]
24. Voller, V.R.; Cross, M. Accurate solutions of moving boundary problems using the enthalpy method. *Int. J. Heat Mass Transfer.* **1981**, *24*, 545–556. [CrossRef]
25. Javierre, E.; Vuik, C.; Vermolen, F.J. A comparison of numerical models for one-dimensional Stefan problems. *J. Comput. Appl. Math.* **2006**, *192*, 445–459. [CrossRef]
26. Caginalp, G. Stefan and Hele-Shaw type models as asymptotic of the phase field equations. *IMA J. Appl. Math. Phys. Rev. A* **1989**, *39*, 5887–5896. [CrossRef] [PubMed]
27. Chen, S.; Merriman, B.; Osher, S.; Smereka, P. A simple level set method for solving Stefan problems. *J. Comput. Phys.* **1997**, *135*, 8–29. [CrossRef]
28. Sussman, M.; Smereka, P.; Osher, S. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.* **1994**, *114*, 146–159. [CrossRef]
29. Tang, S.; Feng, X.; Wu, W.; Xu, H. Physics-informed neural networks combined with polynomial interpolation to solve nonlinear partial differential equations. *Comput. Math. Appl.* **2023**, *132*, 48–62. [CrossRef]

30. Wu, W.; Feng, X.; Xu, H. Improved Deep Neural Networks with Domain Decomposition in Solving Partial Differential Equations. *J. Sci. Comput.* **2022**, *20*, 93. [CrossRef]
31. Peng, P.; Pan, J.; Xu, H. RPINNs: Rectified-physics informed neural networks for solving stationary partial differential equations. *Comput. Fluids* **2022**, *245*, 105583. [CrossRef]
32. Cheng, F.; Xu, H.; Feng, X. Model order reduction method based on (r) POD-ANNs for parameterized time-dependent partial differential equations. *Comput. Fluids* **2022**, *241*, 105481. [CrossRef]
33. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
34. Wang, S.; Perdikaris, P. Deep learning of free boundary and Stefan problems. *J. Comput. Phys.* **2021**, *428*, 109914. [CrossRef]
35. Kitchin, R. Small data in the era of big data. *GeoJournal* **2015**, *80*, 463–475. [CrossRef]
36. Kingma, D.P; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:2014.1412.6980.
37. Byrd, R.H.; Lu, P.; Nocedal, J.; Zhu, C. A limited memory algorithm for bound constrained optimization. *J. Sci. Comput.* **1995**, *16*, 1190–1208. [CrossRef]
38. Bonnerot, R.; Jamet, P.; Jamet, P. Numerical computation of the free boundary for the two-dimensional Stefan problem by space-time finite elements. *J. Comput. Phys.* **1977**, *25*, 163–181. [CrossRef]