Astana IT University

ZUFAR IDOYATOV

Deep learning of free boundary two dimensional direct heat problems

6B06103 — Big Data Analysis

Diploma work

Supervisor
PhD, Associate professor
Kassabek S. A.

Kazakhstan Republic
Nur-Sultan, 2022

CONTENTS

ABSTRACT

Physics-informed neural networks are a groundbreaking technology that was discovered only 5 years ago. They are widely used for solutions of different partial differential equations. Free boundary problems are problems that present great challenge because they are an example of dynamic systems. In this work, I will produce a quantitative assessment of how the multi network architecture based on physics-informed neural networks performs on solving different two-dimensional Stefan problems and I will provide an optimization of the architecture by using learning rate annealing algorithm. Specifically, I will solve unrevised test case of direct two-dimensional one-phase problem as well as introduce solutions to two-dimensional two-phase Stefan problem using physics-informed neural networks, that was not done previously. I will demonstrate how physics-informed neural networks can approximate solutions for two-dimensional two-phase Stefan problems with good accuracy.

DEFINITIONS

Following terms are used in this work:

$\mathbb{R}^d$	Real coordinate space of dimension d.
$A \times B$	Cartesian product that denotes the set of all ordered pairs (a, b) where a is in A and b is in B.
$\boldsymbol{\mathcal{L}}(\boldsymbol{\theta})$	Loss function of the model given the weights $\boldsymbol{\theta}$.
Xavier initialization	Weight initialization method explained in [1].
Adam optimizer	Optimization algorithm for the stochastic gradient descent [2].
Δ	Laplace operator with respect to spatial variables.
Dirichlet boundary condition	Type of boundary condition that specifies the values that a solution must take along the boundary of domain [3].
Neumann boundary condition	Condition that specifies the values of the derivative applied at the boundary of the domain.

DESIGNATIONS AND ABBREVIATIONS

Following designations and abbreviations are used in this work:

PINN	Physics-Informed Neural Network.
ODE	Ordinary differential equation.
PDE	Partial differential equation.
FBP	Free boundary problem.

INTRODUCTION

Work relevance. Deep learning systems are now an indistinguishable component of the process of developing a cutting-edge solution in a variety of fields, including scientific calculations, image classification, natural language processing, and others. Deep learning is now influencing how we build predictive models of compositional systems that simulate complex phenomena, thanks to astounding results obtained in recent research held in various scientific areas [4], [5]. However, in data-rich sectors, the majority of the findings are achieved by utilizing over-parameterized black-box models [6], the nature and interpretability of which are sacrificed in favor of extensibility, estimation capabilities, and computational scalability. The aforementioned is also a case for the numerical solutions of famous free boundary problems - a class of problems explained by PDEs and that are really important to science and engineering as they present complex systems with phase transitions [7]. Recently discovered physics-informed neural networks present an unsupervised solution for the given problems and this work will present a number of observations needed for the development of this technology.

Goal of the work. There is a lack of research on how physics-informed neural networks perform on Stefan problems. In [8] Rohrhofer et. al. explained that there is the superiority of reduced domain methods over standard unmodified physics-informed neural networks, which is mainly because of occurring traps of gradient descent optimizations. We want to investigate the performance of physics-informed neural networks on converging two-dimensional one-phase Stefan problems further by taking the basis of our model from [9].

Research object. Physics-informed neural networks, two-dimensional one-phase Stefan problems.

Novelty. PINNs were first introduced in 2017 [10]. In general, physics-informed neural networks provide an unsupervised data-driven approach to numerically approximating solutions for nonlinear partial differential equations while being constrained by any given law of physics. NanoHUB provides a series of videos for an introduction to PINNs [11]. PINNs were already used for cardiac activation mapping [12], for reconstructing full magnetohydrodynamic solutions [13], for gradient pathologies [14] and a series of other problems regarding PDEs. In [15] Markidis outlines the importance of using transfer learning, and how it is critical to reducing a computational cost. Approximations using PINNs for some Stefan problems were observed in [9], and there is still room for more investigation of the final method.

Research methodology. Using various previously discovered boundary conditions for two-dimensional one-phase Stefan problems this work will construct a physics-informed neural network using Tensorflow backend and python

programming language and asses its performance subjected to different hyper-parameters on the exact solutions.

Practical relevance of the work. Nowadays most machine learning methods are missing the computational science part and are based only on the pattern recognition in the data, which forces them to be weak solvers for numerous partial differential equations problems [16]. Physics-informed neural networks fill the part where we need to use computational science, but there are still many open questions considering issues with their performance, as they can easily fail to learn relevant physics phenomena [17], which means more research has to be done. This work provides more insights on how PINNs behave while training to solve Stefan problems.

Objectives:

1 To discover how physics-informed neural networks converge on unrevised test cases of two-dimensional one-phase Stefan problems.

2 To research how an extension to the architecture of physics-informed neural networks affects convergence.

1 FREE BOUNDARY HEAT PROBLEMS OVERVIEW

In mathematics, free boundary problems are a set of problems defined as a partial differential equation where they have to be solved for both unknown function (in our case temperature) u and for the unknown part of the domain, which is a free boundary Σ itself [7]. An example of free boundary problems is Stefan problems. Free boundary problems and Stefan problems usually classified into two distinct types: direct and inverse. In this paper, we will mainly focus on direct type problems.

1.1 Direct Stefan problems

The direct Stefan problem, in its classical form, entails determining the temperature function in a domain undertaking a phase transition as well as the position of the free boundary defining the transition interface.

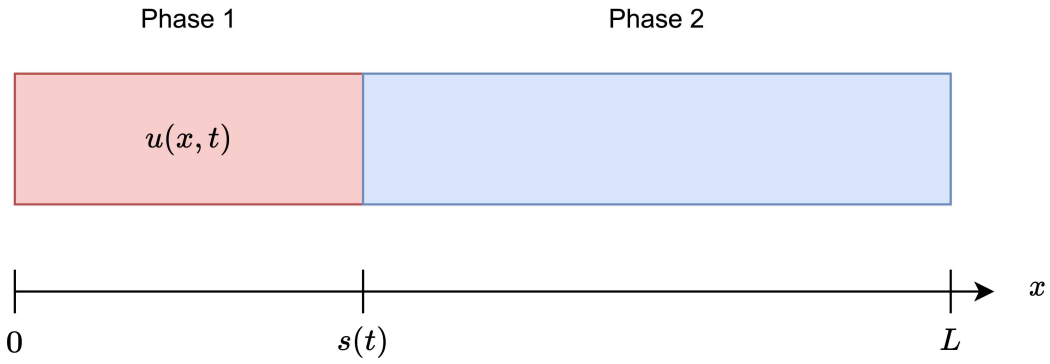


Figure 1.1 – one-dimensional one-phase Stefan problem diagram

Figure 1.1 presents a one-dimensional one-phase Stefan problem where $s(t)$ is a function that defines the position of the free boundary on the number line and $u(x,t)$ is a temperature distribution function. For the quick reference the simplest one-dimensional one-phase direct problem will be explained.

For any $t > 0$, the region $0 \leqslant x < \infty$ will consist of two different phases where the first phase occupies $0 \leqslant x < s(t)$

Let $\Omega = (0, s(t)) \times (0, T)$, $T \in \mathbb{R}$, be the space upon which we will operate, where $u(x,t)$ satisfies the one-dimensional heat equation

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0, \quad (x,t) \in \Omega, \tag{1.1}$$

subject to initial and Neumann boundary conditions

$$u(x,0) = u_0(x), \quad x \in [0, s(0)] \tag{1.2}$$

$$\frac{\partial u}{\partial x}(0,t) = g(t), \quad t \in [0,T]. \tag{1.3}$$

The Dirichlet and Neumann boundary conditions on the free boundary $x = s(t)$ are given by

$$s(0) = s_0 \tag{1.4}$$
$$u(s(t), t) = h_1(t), \quad t \in [0, T] \tag{1.5}$$
$$\frac{\partial u}{\partial x}(s(t), t) = h_2(t), \quad t \in [0, T]. \tag{1.6}$$

Equation 1.4 describes the initial position of the free boundary, while equation 1.5 describes the temperature at the free boundary as both phases can stay together in thermodynamic equilibrium.
Subject to all of the definitions above we need to find $s(t)$ and $u(x, t)$ to solve the one-dimensional one-phase Stefan problem.

1.1.1 Direct two-dimensional one-phase Stefan problems

This is the part where we will explain the main focus of this paper - direct two-dimensional one-phase Stefan problems. Compared to a one-dimensional case, the geometry is rather more complex and some additional notations must be introduced to solve the problem.

Here we present a general formulation for the high-dimensional single phase Stefan problem that was given in [9].

For $0 < t \leqslant T < \infty$, let $\Omega = \bigcup_{0<t\leqslant T}\{(\boldsymbol{x}, t) \in \mathbb{R}^{n+1} : \boldsymbol{x} \in \Omega(t)\}$ be a bounded connected domain in $\mathbb{R}^n$ with boundary $\partial\Omega(t) = \Gamma(t) \bigcup \Sigma(t)$, where $\Gamma = \cup_{0<t<T}\Gamma(t)$ denotes the fixed boundaries and $\Sigma = \cup_{0<t<T}\Sigma(t)$ denotes the free boundaries. Here we assume that Γ is simply connected whereas Σ may be multiply connected. In particular, $\boldsymbol{x} = (x_1, \cdots, x_n)$ denotes a point in $\mathbb{R}^n$ - real n-th dimensional space. The equations describing the Stefan problem at the n-th dimension are as follows:

$$u_t - \Delta u = 0 \quad \text{in } \Omega \tag{1.7}$$
$$u|_{\overline{\Gamma}} = g \tag{1.8}$$
$$u|_{\Omega(0)} = u_0 \tag{1.9}$$
$$u|_{\overline{\Sigma}} = u^* \tag{1.10}$$
$$\frac{\partial u}{\partial n}\Big|_{\overline{\Sigma}} = h \tag{1.11}$$
$$\Sigma(0) = \Sigma_0, \tag{1.12}$$

where n is the normal vector with respect to the space variables in the domain Ω.

As in one-dimensional single phase direct problem, the goal would be to

	Observed	Latent
Direct	u^*, u_0, g, h, Σ_0	$u(\boldsymbol{x}, t), \Sigma(t)$

Table 1.1 – Summary of conditions and the objective of direct type of Stefan problem.

determine the unknown free surface $\Sigma(t)$ as well as the temperature function $u(\boldsymbol{x}, t)$ that satisfies equations 1.7 - 1.9.

2 PHYSICS-INFORMED NEURAL NETWORKS OVERVIEW

The main method used to solve Stefan problems in this paper is physics-informed neural networks. PINNs are used to solve partial differential equations as they are a type of universal function approximators that can use knowledge of physical laws to constrain the algorithm.

Here is the general form for partial differential equations under the framework of PINNs

$$\begin{aligned} &\boldsymbol{u}_t + \mathcal{N}_{\boldsymbol{x}}[\boldsymbol{u}] = 0, \quad \boldsymbol{x} \in \Omega, t \in [0, T] \\ &\boldsymbol{u}(\boldsymbol{x}, 0) = h(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \\ &\boldsymbol{u}(\boldsymbol{x}, t) = g(\boldsymbol{x}, t), \quad t \in [0, T], \quad \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ and $t \in [0, T]$, Ω denotes a bounded domain in $\mathbb{R}^d$ with boundaries $\partial\Omega$, $T > 0$, and $\mathcal{N}_{\boldsymbol{x}}$ is a nonlinear differential operator. Further more, $u(\boldsymbol{x}, t) : \Omega \times [0, T] \to \mathbb{R}$ describes the unknown physical law described by equations 2.1.

Before proceeding to architecture we need to additionally transform Σ to fit the needs of the model, first define $\Sigma(t)$ as

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

where $\Phi(x, y, t) = x - s(y, t)$

Besides, we will add additional notations for the boundary conditions

$$u(x, 0, t) = g_1(x, t) \tag{2.2}$$
$$u(0, y, t) = g_2(y, t) \tag{2.3}$$
$$u(x, 1, t) = g_3(x, t). \tag{2.4}$$

Let $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ be two separate parameter spaces. As in figure 2.1, the PINN is presented as two separate fully-connected deep neural networks $u_{\boldsymbol{\theta}}(x, y, t)$ that approximates $u(x, y, t)$ and $s_{\boldsymbol{\beta}}(y, t)$ for approximating $s(y, t)$. The parameter spaces $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are obtained upon training neural network by minimizing the sum of squared errors loss of equation

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}) = \mathcal{L}_r(\boldsymbol{\theta}) + \mathcal{L}_{u_0}(\boldsymbol{\theta}) + \mathcal{L}_{u_{bc}}(\boldsymbol{\theta}) + \mathcal{L}_{S_{bc}}(\boldsymbol{\theta}, \boldsymbol{\beta}) + \mathcal{L}_{s_{Nc}}(\boldsymbol{\theta}, \boldsymbol{\beta}) + \mathcal{L}_{s_0}(\boldsymbol{\beta}), \tag{2.5}$$

where

$$\mathcal{L}_r(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{\partial u_{\boldsymbol{\theta}}}{\partial t} - \frac{\partial^2 u_{\boldsymbol{\theta}}}{\partial x^2} - \frac{\partial^2 u_{\boldsymbol{\theta}}}{\partial y^2} \right|^2 \tag{2.6}$$

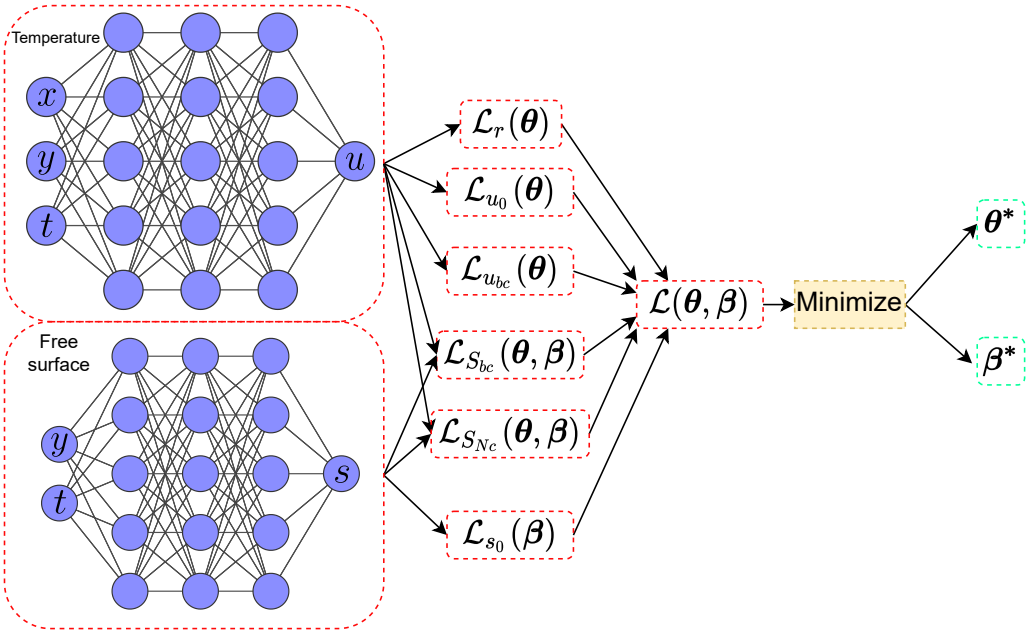


Figure 2.1 – PINN for two-dimensional single phase Stefan problem architecture

$$\mathcal{L}_{u_0}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} |u_{\boldsymbol{\theta}}(x^i, y^i, 0) - u(x^i, y^i, 0)|^2 \tag{2.7}$$

$$\mathcal{L}_{u_{bc1}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} |u_{\boldsymbol{\theta}}(x^i, 0, t^i) - g_1(x^i, t^i)|^2 \tag{2.8}$$

$$\mathcal{L}_{u_{bc2}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} |u_{\boldsymbol{\theta}}(0, y^i, t^i) - g_2(y^i, t^i)|^2 \tag{2.9}$$

$$\mathcal{L}_{u_{bc3}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} |u_{\boldsymbol{\theta}}(x^i, 1, t^i) - g_3(x^i, t^i)|^2 \tag{2.10}$$

$$\mathcal{L}_{u_{bc}}(\boldsymbol{\theta}) = \mathcal{L}_{u_{bc1}}(\boldsymbol{\theta}) + \mathcal{L}_{u_{bc2}}(\boldsymbol{\theta}) + \mathcal{L}_{u_{bc3}}(\boldsymbol{\theta}) \tag{2.11}$$

$$\mathcal{L}_{s_{bc}}(\boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} |u_{\boldsymbol{\theta}}(s_{\boldsymbol{\beta}}(y^i, t^i), y^i, t^i) - u^*|^2 \tag{2.12}$$

$$\mathcal{L}_{s_{Nc}}(\boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} |\frac{\partial u_{\boldsymbol{\theta}}}{\partial \boldsymbol{n}}(s_{\boldsymbol{\beta}}(y^i, t^i), y^i t^i) - h(y^i, t^i)|^2 \tag{2.13}$$

$$\mathcal{L}_{s_0}(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} |s_{\boldsymbol{\beta}}(y^i, 0) - s_0(y^i)|^2. \tag{2.14}$$

where N is the batch size.

2.1 Hyper-parameters settings and computer specifications

Hyperparameter	Value
learning rate	0.001
number of iterations	varied
number of hidden layers	3
number of hidden units	100
batch-size	128
activation function	Hyperbolic Tangent
Initialization method	Glorot
Optimization algorithm	Adam

Table summarizes the hyper-parameter settings that were used for two fully-connected neural networks in the PINN architecture. There is non of the regularization techniques used for the model.

Table 2.1 – Hyper-parameter settings.

In the experiments performed, the number of iterations varied and it was found that after 1000 iterations there are no significant changes, detailed results are shown in subsequent chapters. Besides, in some experiments there was employed a special algorithm that changes how the learning rate behaves to tune total loss function. All of the computing operations were done on Google Colab [18] with NVIDIA T4 GPU, 12Gb of RAM memory and 2 core 4 threads Intel XEON processor.

3 CASE STUDIES

The goal of the following chapter is to solve several two-dimensional one-phase test Stefan problems using PINNs, discover how the different hyper-parameters affect the convergence of the model and to apply optimization algorithm for the weights in the loss function.

3.1 First test problem

Here we will look to a specific direct two-dimensional single phase Stefan example [19] that was also considered and solved in [9] using PINNs.

There follows a definitions of the problem. Let $0 \leqslant t \leqslant T = 1$ and the computational domain given by

$$\Omega(t) = \{(x, y, t) \in \mathbb{R}^3 : 0 < x < s(y, t), 0 < y < 1\} \subset \Omega^*, \tag{3.1}$$

and $\Omega^* = [0, 2.25] \times [0, 1] \times [0, 1]$. As described in table 1.1, we aim to solve equations 1.7 - 1.12 with

$$u_0(x, y) = \exp(-x + \frac{1}{2}y + \frac{1}{2}), \quad x, y \in \Omega(0) \tag{3.2}$$

$$u(s(y, t), y, t) = u^* = 0 \tag{3.3}$$

$$h = \sqrt{\frac{5}{4}} \tag{3.4}$$

$$s(y, 0) = s_0(y) = \frac{1}{2}y + \frac{1}{2}. \tag{3.5}$$

The boundary conditions are given by

$$u(x, 0, t) = g_1(x, t) = \exp(1.25t - x + 1/2) - 1, \quad (x, 0, t) \in \Omega \tag{3.6}$$

$$u(0, y, t) = g_2(y, t) = \exp(1.25t + 0.5y + 1/2) - 1, \quad (0, y, t) \in \Omega \tag{3.7}$$

$$u(x, 1, t) = g_3(x, t) = \exp(1.25t - x + 1), \quad (x, 1, t) \in \Omega. \tag{3.8}$$

The exact solution of this problem is given by

$$u(x, y, t) = \exp(\frac{5}{4}t - x + \frac{1}{2}y + \frac{1}{2}) - 1 \tag{3.9}$$

$$s(y, t) = \frac{1}{2}y + \frac{5}{4}t + \frac{1}{2}. \tag{3.10}$$

To asses model performance and give insightful visualization four representative snapshots of the exact and the predicted temperature distributions $u(x, y, t)$ are shown in figure 3.1. Snapshots are $\delta t = 0.2$ and $t \in [0.2, 0.8]$.

Similar results were shown in [9]. The error for both u and s is of order $O(10^{-2})$ and L^2 error for u is about 2×10^{-2}. Figure 3.3 and table 3.1 shows how the L^2 error for both u and s changes over iterations, from the graph we can see that 1000 iterations is enough for the model to converge, despite [9] using 40000 iterations. For the rest of this chapter we will use 1000 iterations.

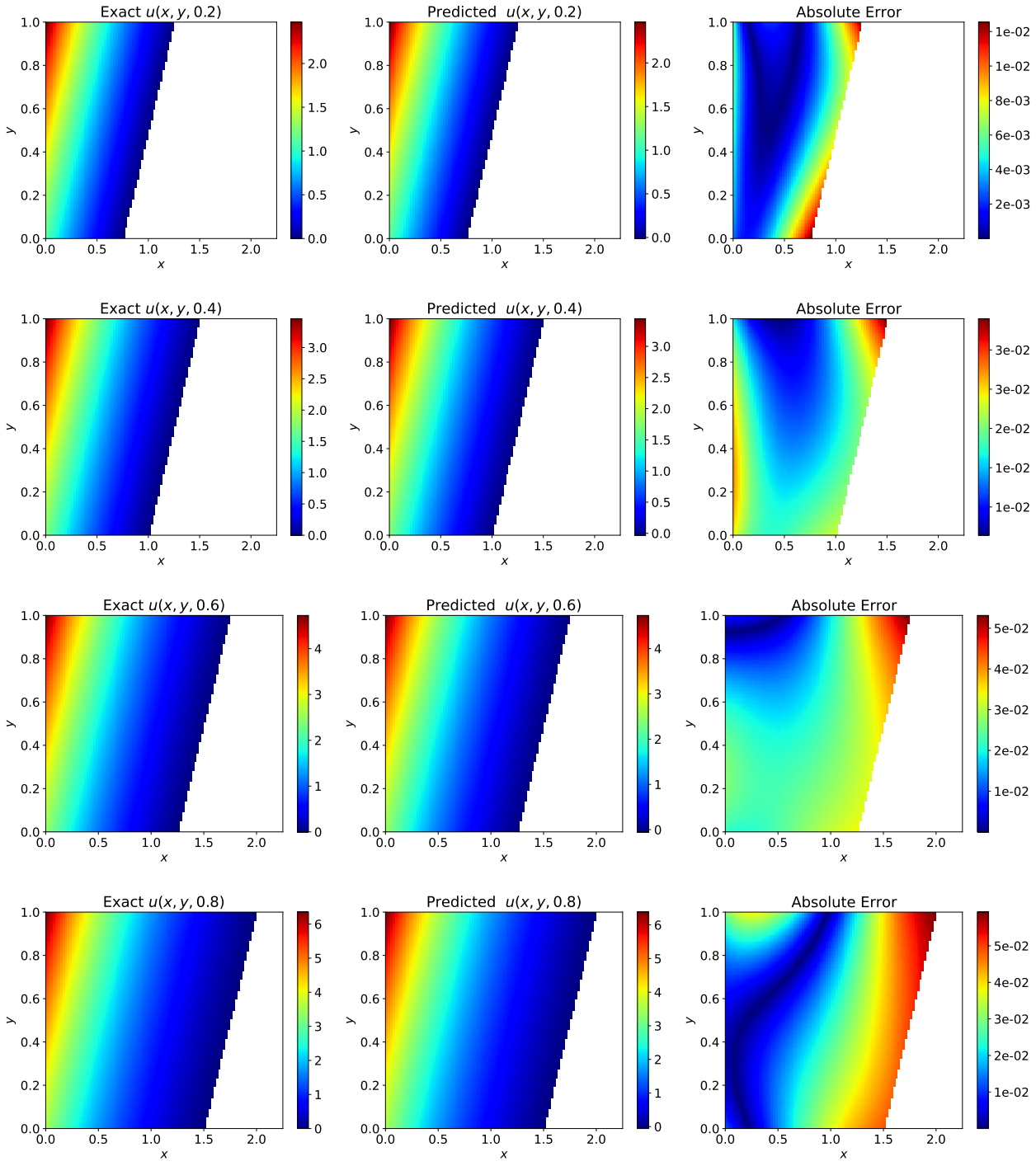


Figure 3.1 – Difference between predicted and exact values corresponding to four different temporal snapshots for the first test problem.

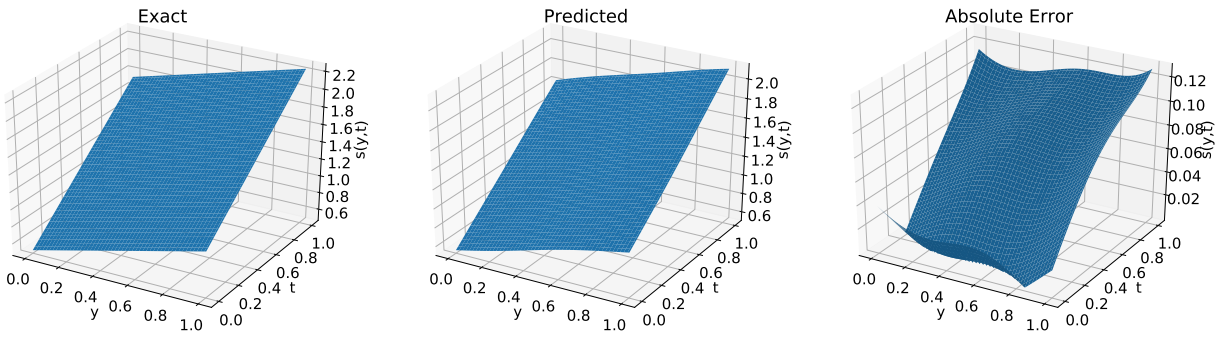
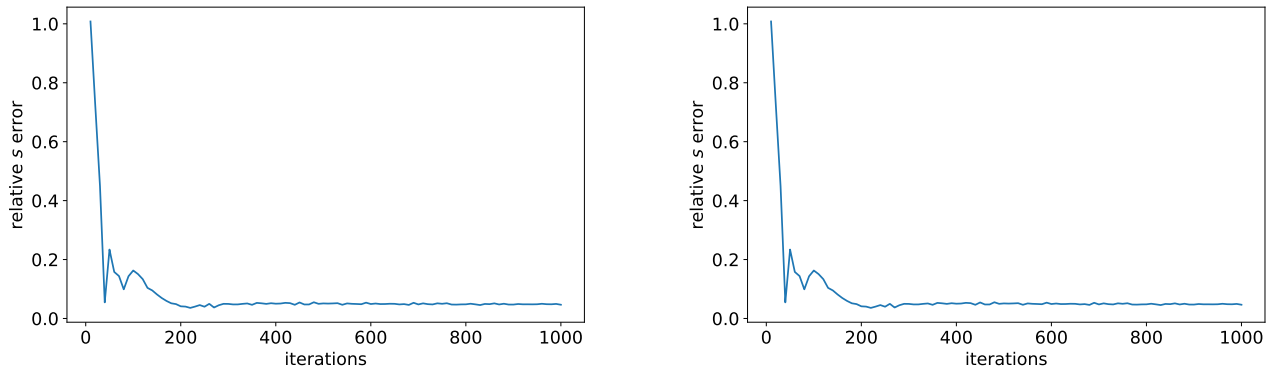


Figure 3.2 – Difference between predicted and exact values for free surface. The relative L^2 error is 4.84×10^{-2}.



(a) relative s error by iterations

(b) relative u error by iterations

Figure 3.3 – Relative L^2 errors by number of iterations

iteration	L^2 error for u	L^2 error for s
100	0.341	0.163
200	0.158	0.041
300	0.058	0.049
400	0.041	0.050
500	0.027	0.051
600	0.026	0.049
700	0.023	0.048
800	0.022	0.048
900	0.019	0.047
1000	0.020	0.047

Table 3.1 – Relative L^2 error by number of iterations.

3.2 Second test problem

Here we will look to another direct two-dimensional single phase Stefan example [20] that was not considered and solved in the paper [9].

Similarly as in previous problem let $0 \leqslant t \leqslant T = 1$ and the computational domain $\Omega(t)$ stay with same definition as in equation 3.1. Now the $\Omega^* = [0, 0.95] \times [0, 1] \times [0, 1]$. Here are the conditions defined for this problem

$$u_0(x, y) = \frac{(y+1)^2}{20} + x, \quad x, y \in \Omega(0) \tag{3.11}$$

$$u(s(y, t), y, t) = u^* = 1 \tag{3.12}$$

$$h = \sqrt{\left(\frac{y+1}{10}\right)^2 + 1} \tag{3.13}$$

$$s(y, 0) = s_0(y) = 1 - \frac{(y+1)^2}{20}. \tag{3.14}$$

The boundary conditions are given by

$$g_1(x, t) = \frac{1}{20} + x + \frac{t}{10}, \quad (x, 0, t) \in \Omega \tag{3.15}$$

$$g_2(y, t) = \frac{(y+1)^2}{20} + \frac{t}{10}, \quad (0, y, t) \in \Omega \tag{3.16}$$

$$g_3(x, t) = \frac{1}{5} + x + \frac{t}{10}, \quad (x, 1, t) \in \Omega. \tag{3.17}$$

And the exact solution is given by

$$u(x, y, t) = \frac{(y+1)^2}{20} + x + \frac{t}{10} \tag{3.18}$$

$$s(y, t) = 1 - \frac{t}{10} - \frac{(y+1)^2}{20}. \tag{3.19}$$

Generally, this problem is considered easier for numerical methods to solve it as it involves polynomial functions which are much lighter for convergence compared to exponential functions as in first test problem. Still, we want to look at how PINN behave while solving this problem. From figures 3.4 and 3.5 we clearly can see that we achieved good results for the prediction of the model which shows that PINNs approximate well this Stefan problem. Figure 3.6, again, approves that 1000 iterations is enough for model to converge.

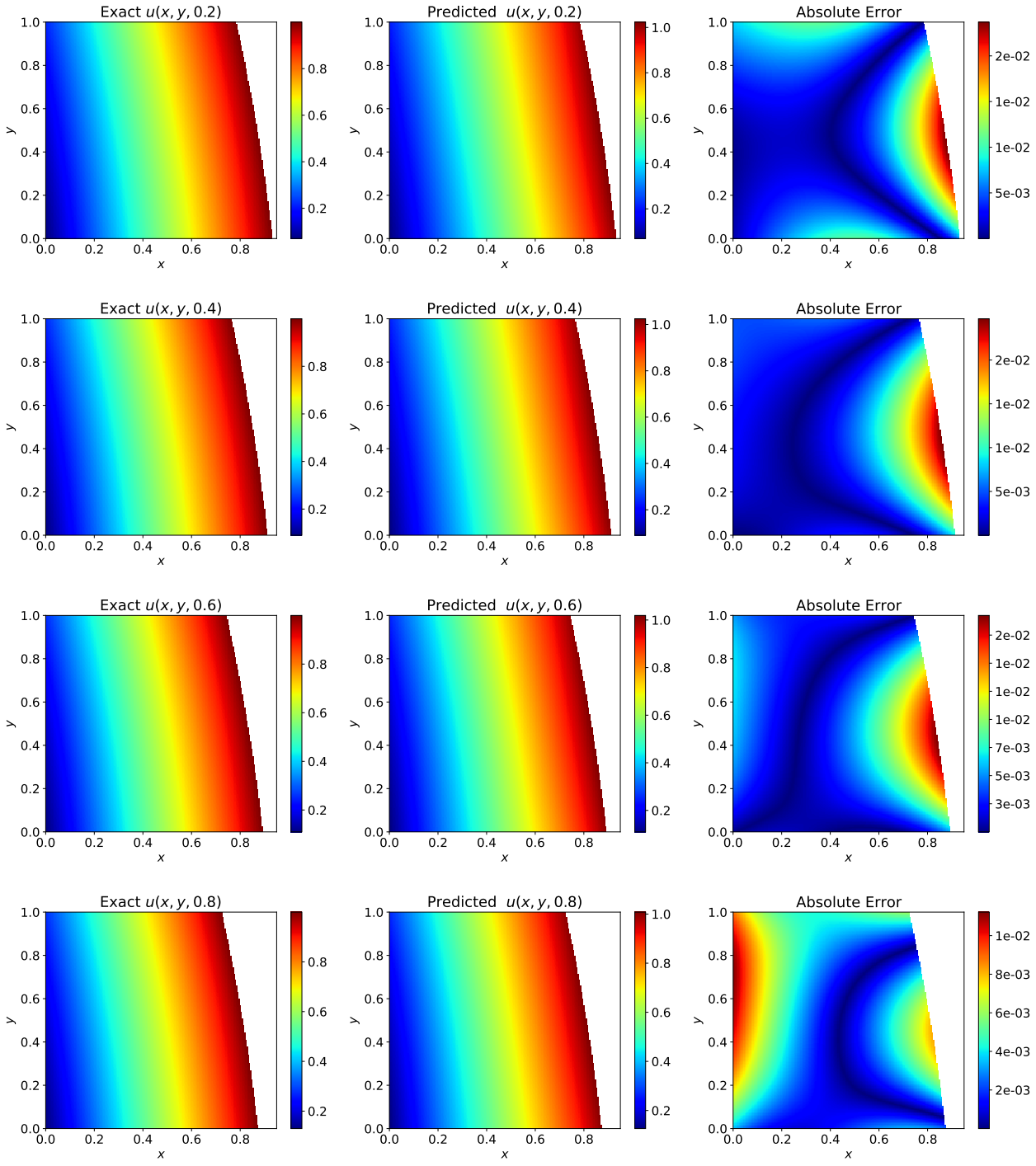


Figure 3.4 – Difference between predicted and exact values corresponding to four different temporal snapshots for the second test problem.

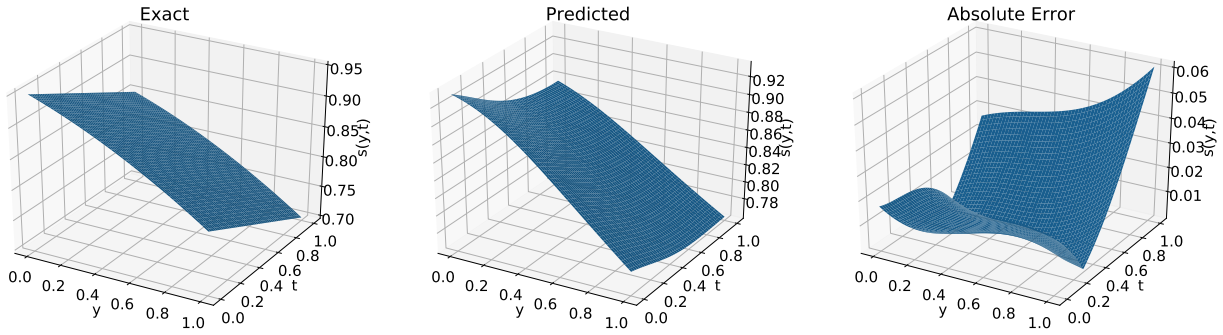
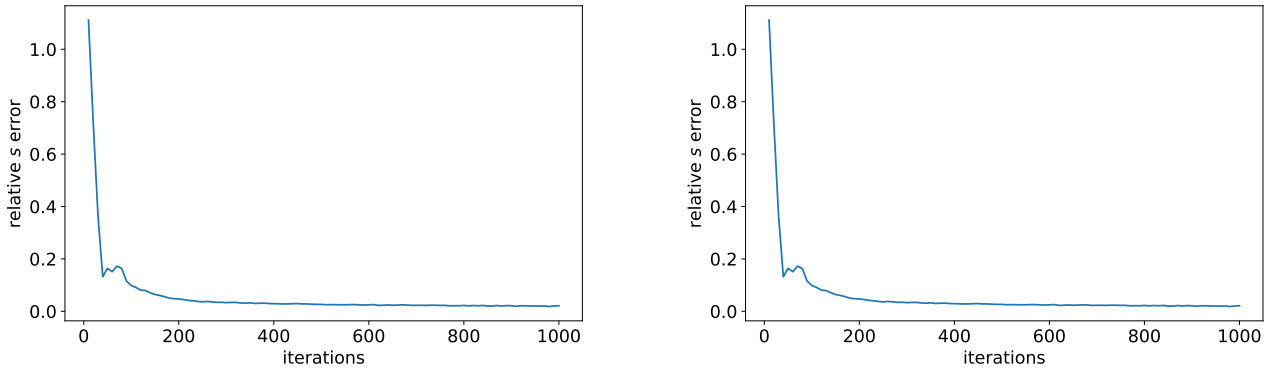


Figure 3.5 – Comparison of the predicted and exact free surface for the second test problem. The relative L^2 error is 2.02×10^{-2}.



(a) relative s error by iterations

(b) relative u error by iterations

Figure 3.6 – Relative L^2 errors by number of iterations

3.3 Implementing learning rate annealing algorithm

In [9] Wang et. al. referenced the algorithm introduced in [14] and used it only for inverse type II Stefan problems. For this section we want to implement this algorithm in the models and see how it improves the convergence of PINNs for both first and second two-dimensional one-phase test Stefan problems.

Here we present an algorithm from the original paper

Algorithm 1: Learning rate annealing for PINNs

Suppose $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a PINN with parameter space $\boldsymbol{\theta}$ and defined loss function $\mathcal{L}(\boldsymbol{\theta})$ where

$$\mathcal{L}(\boldsymbol{\theta}) := \mathcal{L}_r(\boldsymbol{\theta}) + \sum_{i=1}^{M} \lambda_i \mathcal{L}_i(\boldsymbol{\theta}),$$

where $\mathcal{L}_r(\boldsymbol{\theta})$ is the partial differential equation residual loss, the $\mathcal{L}_i(\boldsymbol{\theta}) \in \{\mathcal{L}_{u_0}(\boldsymbol{\theta}), \mathcal{L}_{u_{bc}}(\boldsymbol{\theta}), \mathcal{L}_{S_{cs}}(\boldsymbol{\theta})\}$, and $\mathcal{L}_{S_{cs}}(\boldsymbol{\theta}) = \mathcal{L}_{S_{bc}}(\boldsymbol{\theta}) + \mathcal{L}_{S_0}(\boldsymbol{\theta}) + \mathcal{L}_{S_{Nc}}(\boldsymbol{\theta})$ and $\lambda_i = 1$, $i = 1, \ldots, M$ are variables that will tuned - weights that are needed to manage the interaction of the various loss terms. By using K steps of a gradient descent algorithm to update the parameters $\boldsymbol{\theta}$ as:

for $n = 1, \ldots, K$ **do**

(a) Recalculate $\hat{\lambda}_i$ by

$$\hat{\lambda}_i = \frac{\max_{\boldsymbol{\theta}}\{|\nabla_{\boldsymbol{\theta}} \mathcal{L}_r(\boldsymbol{\theta_n})|\}}{\overline{|\nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta_n})|}}, \quad i = 1, \ldots, M, \tag{1}$$

where $\overline{|\nabla_{\boldsymbol{\theta_n}} \mathcal{L}_i(\boldsymbol{\theta_n})|}$ is average value of $|\nabla_{\boldsymbol{\theta_n}} \mathcal{L}_i(\boldsymbol{\theta_n})|$ with respect to parameter space $\boldsymbol{\theta}$.

(b) Update the weights λ_i using a moving average of the form

$$\lambda_i = (1 - \alpha)\lambda_i + \alpha \hat{\lambda}_i, \quad i = 1, \ldots, M. \tag{2}$$

(c) Update the parameters $\boldsymbol{\theta}$ via gradient descent

$$\boldsymbol{\theta_{n+1}} = \boldsymbol{\theta_n} - \eta \nabla_{\boldsymbol{\theta_n}} \mathcal{L}_r(\boldsymbol{\theta_n}) - \eta \sum_{i=1}^{M} \lambda_i \nabla_{\theta} \mathcal{L}_i(\boldsymbol{\theta_n}) \tag{3}$$

end

Where $\eta = 10^{-3}$ is an initial learning rate and $\alpha = 0.9$ is a moving average constant, and it is possible to play with different values of η and α.

It has to be noted that this particular algorithm could be changed and the way how different weights are setup are up to final user. Furthermore, one downside of this algorithm is that it takes significantly more time for the model to train as it requires to compute gradients for each loss function at each iteration.

3.3.1 Learning rate annealing for the first test problem

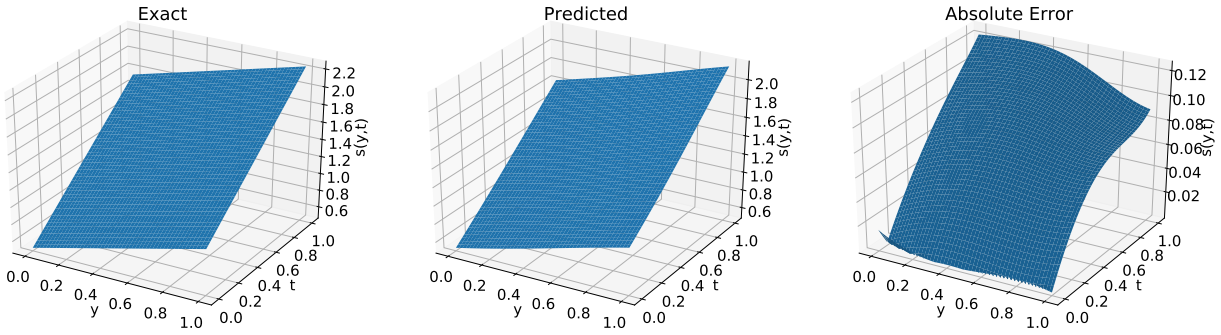
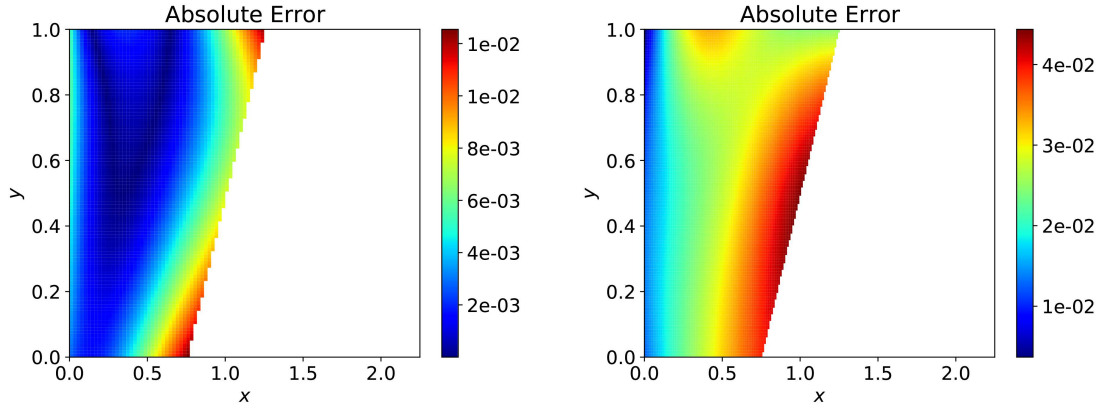


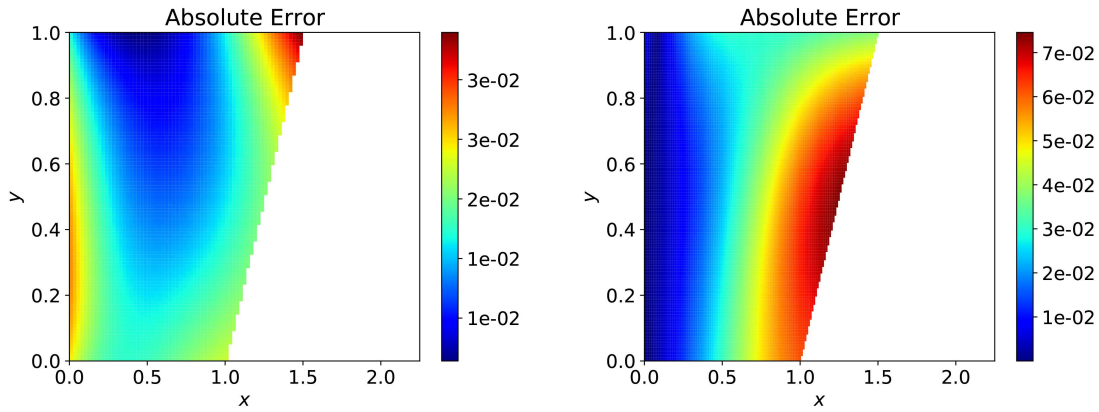
Figure 3.7 – Comparison of the predicted and exact free surface for the first test problem with learning rate annealing. The relative L^2 error is 5.32×10^{-2}.



(a) without annealing (b) with annealing

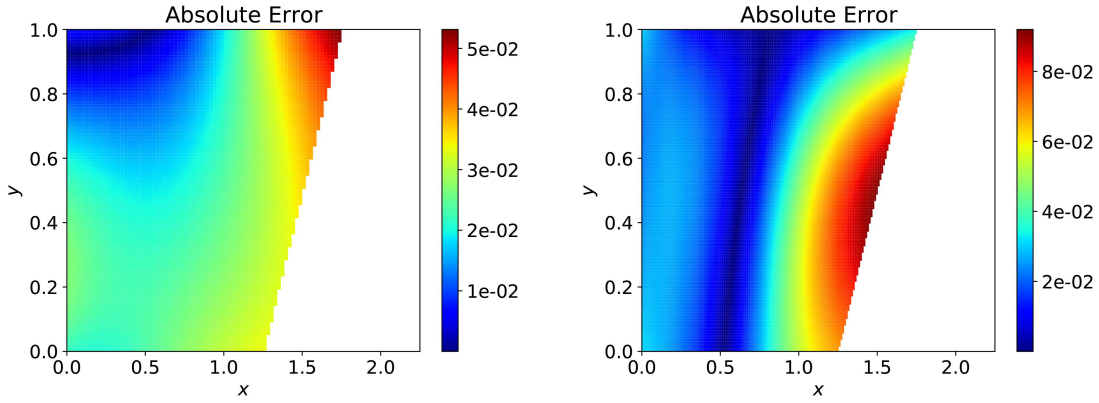
Figure 3.8 – Absolute error in $u(x, y, t = 0.2)$

From figures 3.8 - 3.11 we could see that new algorithm negatively affects predictions for the temperature distribution for the first test problem and from figure 3.7 we could see that it does same for the free boundary prediction as relative error is now 5.32×10^{-2} instead of 4.84×10^{-2}.



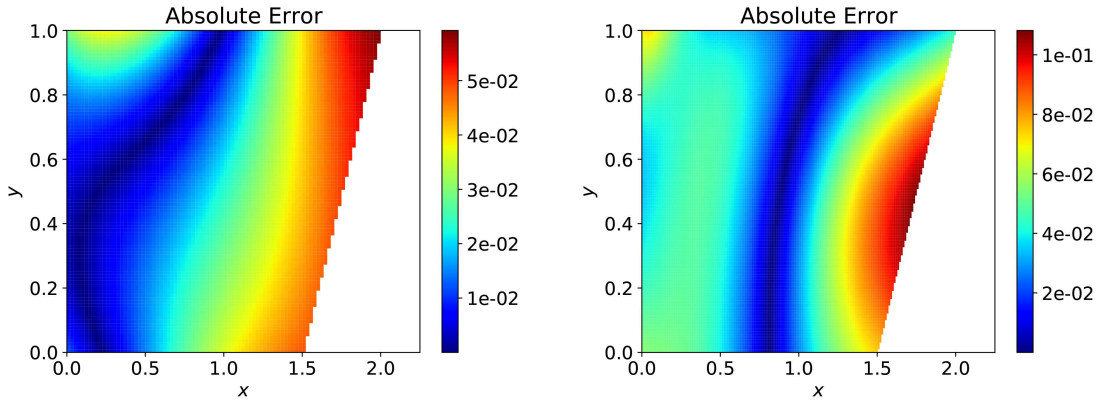
(a) without annealing (b) with annealing

Figure 3.9 – Absolute error in $u(x, y, t = 0.4)$



(a) without annealing (b) with annealing

Figure 3.10 – Absolute error in $u(x, y, t = 0.6)$



(a) without annealing (b) with annealing

Figure 3.11 – Absolute error in $u(x, y, t = 0.8)$

3.3.2 Learning rate annealing for the second test problem

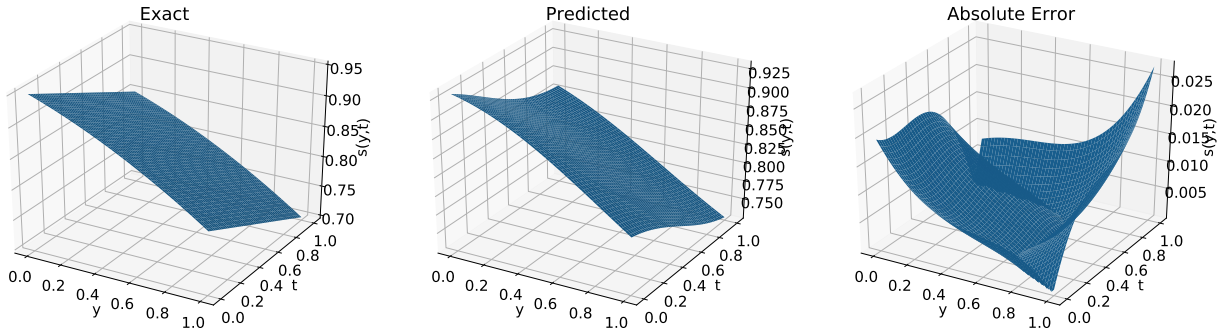
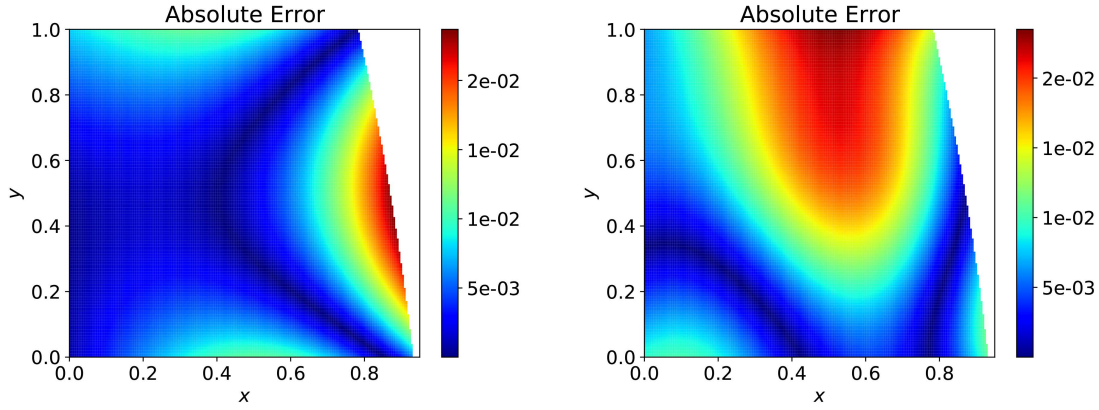


Figure 3.12 – Difference between predicted and exact values for free surface for the second test problem with learning rate annealing. The relative L^2 error is 1.02×10^{-2}.

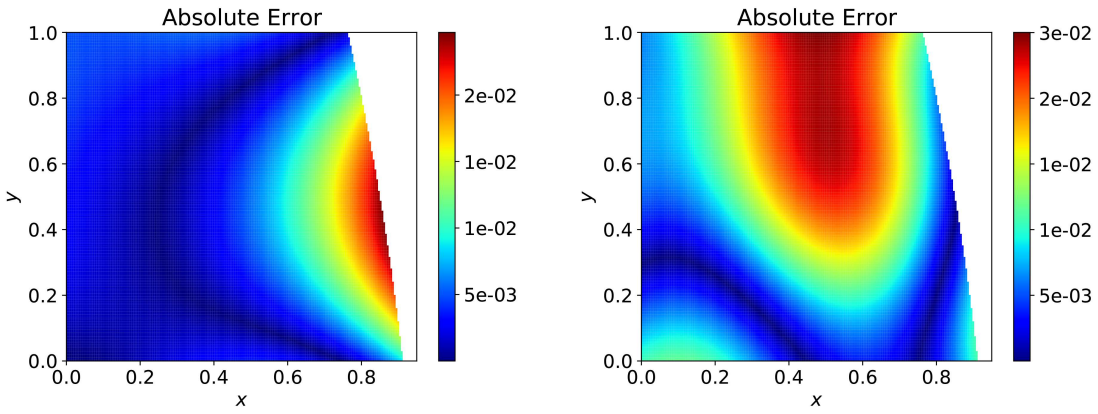


(a) without annealing (b) with annealing

Figure 3.13 – Absolute error in $u(x, y, t = 0.2)$

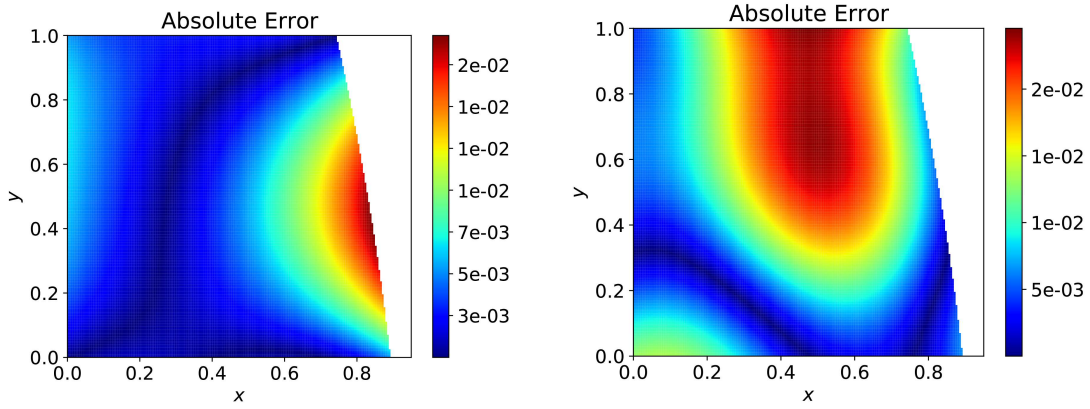
From figures 3.13 - 3.16 we could see that new algorithm is slightly worse in predictions for the temperature distribution for the second test problem. However, from the figure 3.12 the relative error is two times smaller for the new algorithm, this could imply that the new algorithm could produce better results.

It has to be noted that original algorithm was intended for architecture with one neural network and algorithm in this work is an extension as it utilizes parameter spaces of both neural networks. This could be the reason why the algorithm might perform worse on some of the problems.



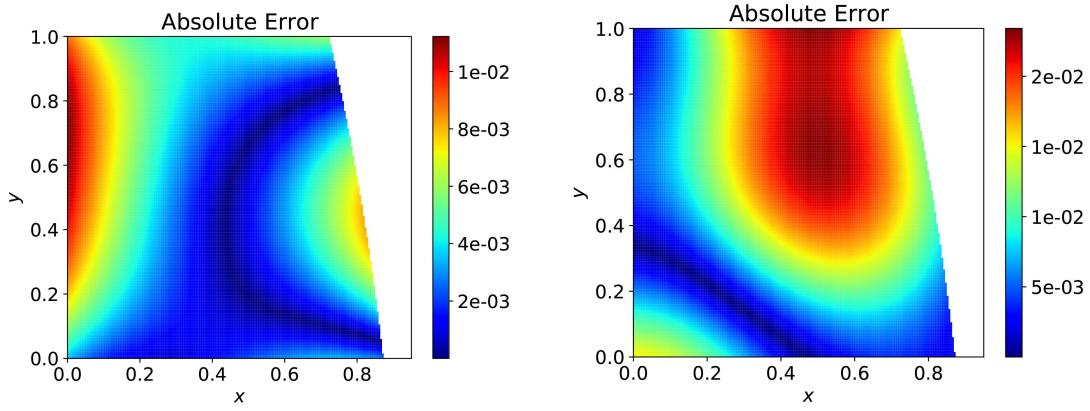
(a) without annealing (b) with annealing

Figure 3.14 – Absolute error in $u(x, y, t = 0.4)$



(a) without annealing (b) with annealing

Figure 3.15 – Absolute error in $u(x, y, t = 0.6)$



(a) without annealing (b) with annealing

Figure 3.16 – Absolute error in $u(x, y, t = 0.8)$

3.4 Two dimensional two phase direct Stefan problem

One shortcoming of the paper by Wang et. al. [9] is that it remained the open question: could the architecture they provided solve two-dimensional two-phase direct Stefan problems. By extending their work we will solve this problem and asses the performance of this algorithm. Before we proceed we need to introduce more definitions for two-phase problems and redefine loss functions of the PINN to suit the two-phase Stefan problem.

3.4.1 Mathematical formulation

In this subsection we will extend notations given in section 1.1 to two phase problems. Let first phase region be defined as

$$\Omega_1(t) = \{(x, y) \in \mathbb{R} | 0 < x < s(y, t), 0 < y < 1\},$$

and the second phase region be defined as

$$\Omega_2(t) = \{(x, y) \in \mathbb{R} | s(y, t) < x < l, 0 < y < 1\},$$

where $s(y, t) \in (0, l)$. The heat equation are then as follows:

$$\frac{\partial u_1}{\partial t} - \alpha_1 \Delta u_1 = 0 \quad \text{in } \Omega_1, \tag{3.20}$$

$$\frac{\partial u_2}{\partial t} - \alpha_2 \Delta u_2 = 0 \quad \text{in } \Omega_2, \tag{3.21}$$

where $\alpha_1, \alpha_2 > 0$ are thermal diffusivity constants for first and second phase conductors respectively. The boundary conditions along $\overline{\Sigma}$ are given by

$$u_1|_{\overline{\Sigma}} = u_2|_{\overline{\Sigma}} = u^*, \tag{3.22}$$

$$\frac{\partial s}{\partial t}|_{\overline{\Sigma}} = -K_1 \frac{\partial u_1}{\partial \boldsymbol{n}} + K_2 \frac{\partial u_2}{\partial \boldsymbol{n}} \tag{3.23}$$

where $K_1, K_2 > 0$.

3.4.2 Architecture

In original work of Wang et. al. [9] used one neural network $u_{\boldsymbol{\theta}}(x, y, t)$ to output values for both u_1 and u_2, we will do the same. Note that for two dimensional case all fixed boundaries conditions are given by

$$g_1(x, t) = \begin{cases} u_1(x, 0, t), & x < s(0, t) \\ u_2(x, 0, t), & x \geqslant s(0, t) \end{cases} \tag{3.24}$$

$$u_1(0, y, t) = g_2(y, t) \tag{3.25}$$

$$g_3(x, t) = \begin{cases} u_1(x, 1, t) & x < s(1, t) \\ u_2(x, 1, t), & x \geqslant s(1, t) \end{cases} \tag{3.26}$$

$$u_2(l, y, t) = g_4(y, t) \tag{3.27}$$

note that we could use following definition for the neural network that applies to whole domain:

$$u_{\boldsymbol{\theta}}(x, y, t) = u_{\boldsymbol{\theta}}^{(1)}(x, y, t)\mathbf{1}_{(0, s_{\boldsymbol{\beta}}(y,t))}(x) + u_{\boldsymbol{\theta}}^{(2)}(x, y, t)\mathbf{1}_{(s_{\boldsymbol{\beta}}(y,t), l)}(x), \tag{3.28}$$

where $u_{\boldsymbol{\theta}}^{(1)}(x, y, t)$ and $u_{\boldsymbol{\theta}}^{(2)}(x, y, t)$ are outputs of the last layer of the neural network. Similarly we could define exact solution:

$$u(x, y, t) = u_1(x, y, t)\mathbf{1}_{(0, s(y,t))}(x) + u_2(x, y, t)\mathbf{1}_{(s(y,t), l)}(x) \tag{3.29}$$

where we say that $\mathbf{1}_{(a,b)}(x)$ denotes a special indication function that gives a value of 1 if $x \in (a, b)$ and 0 otherwise.
Finally we could start defining loss functions that are needed to minimize parameter spaces $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$.

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}) = \sum_{k=1}^{2} \left[\mathcal{L}_r^{(k)}(\boldsymbol{\theta}) + \mathcal{L}_{u_0}^{(k)}(\boldsymbol{\theta}) + \mathcal{L}_{s_{bc}}^{(k)}(\boldsymbol{\theta}, \boldsymbol{\beta}) \right] + \mathcal{L}_{u_{bc}}(\boldsymbol{\theta}) + \mathcal{L}_{s_{Nc}}(\boldsymbol{\theta}, \boldsymbol{\beta}) + \mathcal{L}_{s_0}(\boldsymbol{\beta}), \tag{3.30}$$

where,

$$\mathcal{L}_r^{(k)}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} |\frac{\partial u_{\boldsymbol{\theta}}^{(k)}}{\partial t}(x^i, y^i, t^i) - \alpha_i (\frac{\partial^2 u_{\boldsymbol{\theta}}^{(k)}}{\partial x^2}(x^i, y^i, t^i) + \frac{\partial^2 u_{\boldsymbol{\theta}}^{(k)}}{\partial y^2}(x^i, y^i, t^i))|^2 \tag{3.31}$$

$$\mathcal{L}_{u_0}^{(k)}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} |u_{\boldsymbol{\theta}}^{(k)}(x^i, y^i, 0) - u^{(k)}(x^i, y^i, 0)|^2 \tag{3.32}$$

$$\mathcal{L}_{s_{bc}}^{(k)}(\boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} |u_{\boldsymbol{\theta}}^{(k)}(s_{\boldsymbol{\beta}}(y^i, t^i), y^i, t^i) - u^*|^2, \tag{3.33}$$

for $k = 1, 2$ and

$$\mathcal{L}_{u_{bc1}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} |u_{\boldsymbol{\theta}}(x^i, 0, t^i) - g_1(x^i, t^i)|^2 \tag{3.34}$$

$$\mathcal{L}_{u_{bc2}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} |u_{\boldsymbol{\theta}}(0, y^i, t^i) - g_2(y^i, t^i)|^2 \tag{3.35}$$

$$\mathcal{L}_{u_{bc3}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} |u_{\boldsymbol{\theta}}(x^i, 1, t^i) - g_3(x^i, t^i)|^2 \tag{3.36}$$

$$\mathcal{L}_{u_{bc4}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} |u_{\boldsymbol{\theta}}(l, y^i, t^i) - g_4(y^i, t^i)|^2 \tag{3.37}$$

$$\mathcal{L}_{u_{bc}}(\boldsymbol{\theta}) = \mathcal{L}_{u_{bc1}}(\boldsymbol{\theta}) + \mathcal{L}_{u_{bc2}}(\boldsymbol{\theta}) + \mathcal{L}_{u_{bc3}}(\boldsymbol{\theta}) + \mathcal{L}_{u_{bc4}}(\boldsymbol{\theta}), \tag{3.38}$$

and

$$\mathcal{L}_{S_{Nc}}(\boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} |\frac{\partial s_{\boldsymbol{\beta}}}{\partial t} - K_1(\frac{\partial u_{\boldsymbol{\theta}}^{(1)}}{\partial x} - \frac{\partial u_{\boldsymbol{\theta}}^{(1)}}{\partial y}\frac{\partial s_{\boldsymbol{\beta}}}{\partial y}) + K_2(\frac{\partial u_{\boldsymbol{\theta}}^{(2)}}{\partial x} - \frac{\partial u_{\boldsymbol{\theta}}^{(2)}}{\partial y}\frac{\partial s_{\boldsymbol{\beta}}}{\partial y})|^2, \tag{3.39}$$

where for $\mathcal{L}_{S_{Nc}}(\boldsymbol{\theta}, \boldsymbol{\beta})$, $(x^i, y^i, t^i) \in \overline{\Sigma}$ and $\mathcal{L}_{s_0}(\boldsymbol{\beta})$ is same as before.

3.4.3 First test problem

Here are the problems input data that was given in [21]. First, let $\Omega^* = [0,3] \times [0,1] \times [0,1] \supset \Omega$ be an artificial computational domain. Then the conditions are given by

$$u_1(x, y, 0) = \exp(-x + \frac{y+1}{2}) - 1, \quad x, y \in \Omega_1(0) \tag{3.40}$$

$$u_2(x, y, 0) = \exp(-2x + y + 1) - 1, \quad x, y \in \Omega_2(0) \tag{3.41}$$

$$u^* = 0 \tag{3.42}$$

$$h_1 = \sqrt{\frac{5}{4}}, \quad h_2 = \sqrt{5} \tag{3.43}$$

$$s_0(y) = \frac{y}{2} + \frac{1}{2} \tag{3.44}$$

and the fixed boundary conditions are given by

$$g_1(x,t) = \begin{cases} \exp(-x + 0.5 + \frac{5t}{4}) - 1, & x < s(0,t) \\ \exp(-2x + 1 + \frac{5t}{2}) - 1, & x \geqslant s(0,t) \end{cases}, \quad (x, 0, t) \in \Omega \tag{3.45}$$

$$g_2(y,t) = \exp(y + 1 + \frac{5t}{4}) - 1, \quad (0, y, t) \in \Omega \tag{3.46}$$

$$g_3(x,t) = \begin{cases} \exp(-x + 1 + \frac{5t}{4}) - 1, & x < s(1,t) \\ \exp(-2x + 2 + \frac{5t}{2}) - 1, & x \geqslant s(1,t) \end{cases}, \quad (x, 1, t) \in \Omega \tag{3.47}$$

$$g_4(y,t) = \exp(y - 5 + \frac{5t}{2}) - 1, \quad (3, y, t) \in \Omega \tag{3.48}$$

. Here is the exact solution

$$u_1(x, y, t) = \exp(-x + \frac{y+1}{2} + \frac{5t}{4}) - 1, (x, y, t) \in \Omega_1 \tag{3.49}$$

$$u_2(x, y, t) = \exp(-2x + y + 1 + \frac{5t}{2}) - 1, (x, y, t) \in \Omega_2 \tag{3.50}$$

$$s(y,t) = \frac{y+1}{2} + \frac{5t}{4}, \tag{3.51}$$

and finally $\alpha_1 = 1, \alpha_2 = 0.5, K_1 = \frac{7}{2\sqrt{5}}, K_2 = \frac{3}{\sqrt{5}}$.

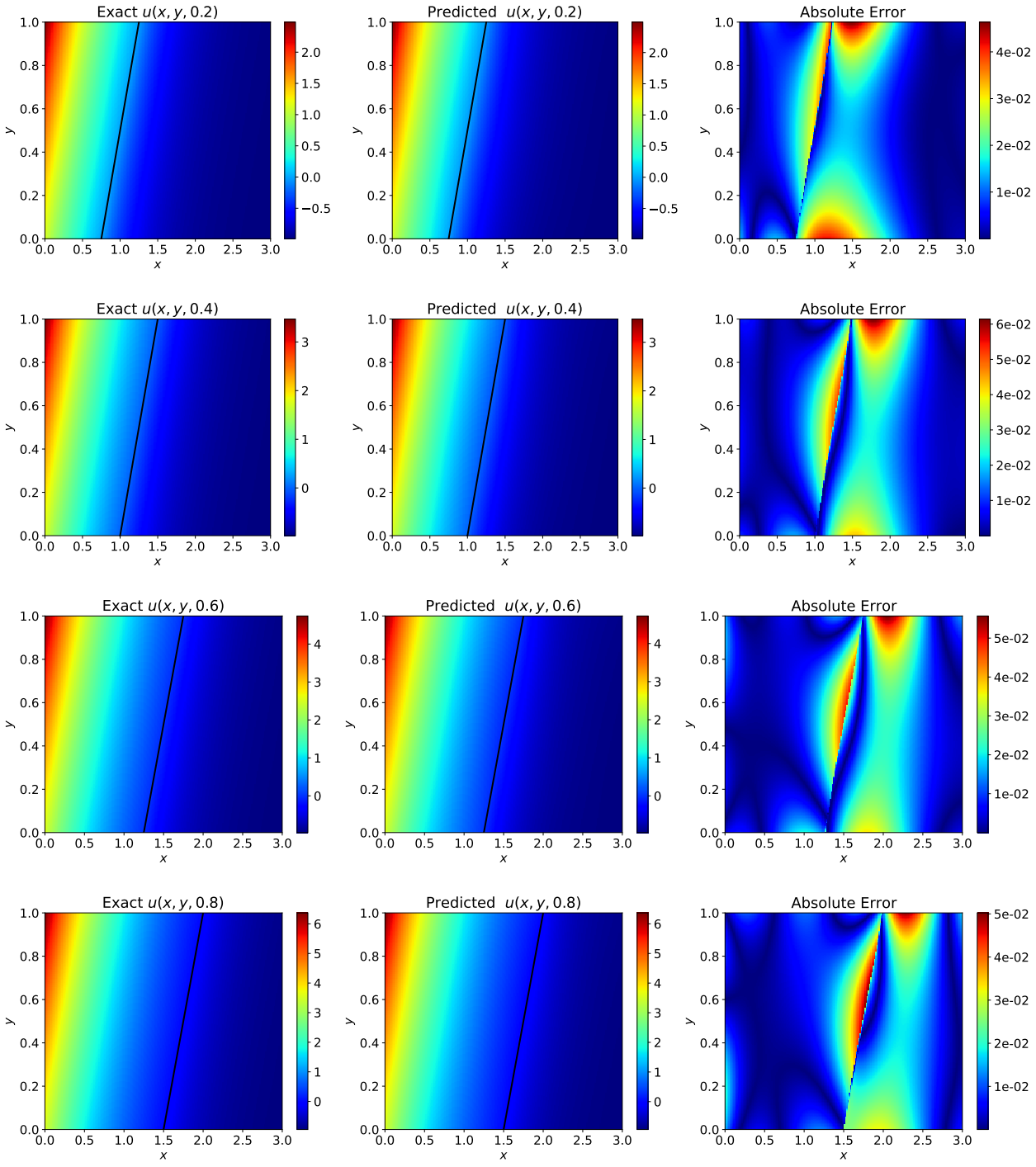


Figure 3.17 – Difference between predicted and exact values corresponding to four different temporal snapshots for the two-dimensional two-phase problem.

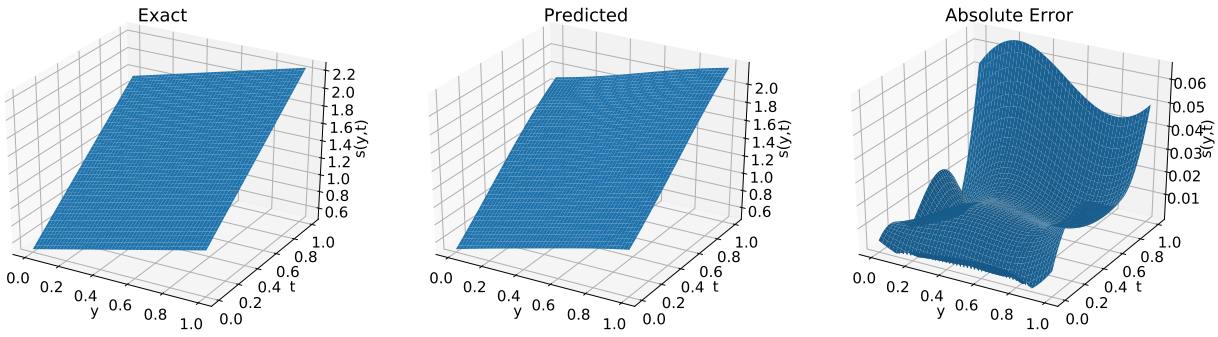


Figure 3.18 – Comparison of the predicted and exact free surface for the two-dimensional two-phase problem. The relative L^2 error is 1.31×10^{-2}.

As we see from the figure 3.17 we are getting quite accurate results with absolute error being of order 10^{-2}. Even more astonishing result we can see in 3.18, the free boundary relative error is even less than in one-phase problem, however, that might be due to random behavior of Xavier initialization function.

CONCLUSION

Free boundary problems could be challenging tasks to approximate with little margin of error. In this work we have reflected on the framework provided by [9] and extended the functionality of the physics-informed neural network. Results acquired:

- Found out that the optimal number of iterations for PINNs to converge is about 1000 instead of 40000.

- Solved using PINN two-dimensional one-phase Stefan problem that was introduced in [20] and dismissed in [9].

- Tested a learning rate annealing algorithm for the PINNs that was first introduced in [14] on how it performs on two-dimensional one-phase Stefan problems.

- Solved using PINN two-dimensional two-phase Stefan problem and showed the validity of PINNs for approximating solutions for two-dimensional two-phase Stefan problems.

Besides, this work proposes to further investigate on how an improved fully-connected neural architecture from [14] will improve the convergence of PINN model for two-dimensional Stefan problems. As another proposal, one might look at how curriculum learning [22] could improve rate at which the model converges. Krishnapriyan et. al. already have shown that curriculum learning provides a significant boost of performance for the PINNs on PDE [17].

ACKNOWLEDGEMENTS

I would like to acknowledge some people without whom the completion of this work wouldn't be possible. I extend my deepest gratitude to Samat Kassabek, my supervisor, for providing scientific background and various testing problems for my diploma work. I would also like to thank my fellow group mates Kamilla Ten, Diana Dalenova, Yedige Ashmet, Bakhtiyar Mautov and Elina Amurlayeva for collaboration, peer review and emotional support.

BIBLIOGRAPHY

1 *Glorot, Xavier*. Understanding the difficulty of training deep feedforward neural networks / Xavier Glorot, Yoshua Bengio // Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics / Ed. by Yee Whye Teh, Mike Titterington. — Vol. 9 of *Proceedings of Machine Learning Research*. — Chia Laguna Resort, Sardinia, Italy: PMLR, 2010. — 13–15 May. — Pp. 249–256. https://proceedings.mlr.press/v9/glorot10a.html.

2 *Kingma, Diederik P.* Adam: A Method for Stochastic Optimization. — 2014. https://doi.org/10.48550/arxiv.1412.6980.

3 *Cheng, Alexander H.-D.* Heritage and early history of the boundary element method / Alexander H.-D. Cheng, Daisy T. Cheng // *Engineering Analysis with Boundary Elements*. — 2005. — Vol. 29, no. 3. — Pp. 268–302. https://www.sciencedirect.com/science/article/pii/S0955799705000020.

4 *Ramesh, Aditya*. Zero-Shot Text-to-Image Generation. — 2021. https://arxiv.org/abs/2102.12092.

5 *Hudson, Drew A.* Generative Adversarial Transformers. — 2021. https://arxiv.org/abs/2103.01209.

6 *Brown, Tom B.* Language Models are Few-Shot Learners. — 2020. https://arxiv.org/abs/2005.14165.

7 *Chen, Gui Qiang*. Free boundary problems: the forefront of current and future developments / Gui Qiang Chen, Henrik Shahgholian, Juan Luis Vazquez // *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*. — 2015. — 9. — Vol. 373. https://royalsocietypublishing.org/doi/full/10.1098/rsta.2014.0285.

8 *Rohrhofer, Franz M.* Understanding the Difficulty of Training Physics-Informed Neural Networks on Dynamical Systems. — 2022. https://arxiv.org/abs/2203.13648.

9 *Wang, Sifan*. Deep learning of free boundary and Stefan problems / Sifan Wang, Paris Perdikaris // *Journal of Computational Physics*. — 2021. — mar. — Vol. 428. — P. 109914. https://doi.org/10.1016%2Fj.jcp.2020.109914.

10 *Raissi, Maziar*. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. — 2017.

11 *Hans, Atharva*. A Hands-on Introduction to Physics-Informed Neural Networks. — 2021. — May. https://nanohub.org/resources/handsonpinns.

12 Physics-Informed Neural Networks for Cardiac Activation Mapping / Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris et al. // *Frontiers in Physics*. — 2020. — Vol. 8. https://www.frontiersin.org/article/10.3389/fphy.2020.00042.

13 *Bard, C.* Neural Network Reconstruction of Plasma Space-Time / C. Bard, J.C. Dorelli // *Frontiers in Astronomy and Space Sciences*. — 2021. — Vol. 8. https://www.frontiersin.org/article/10.3389/fspas.2021.732275.

14 *Wang, Sifan.* Understanding and mitigating gradient pathologies in physics-informed neural networks. — 2020. https://arxiv.org/abs/2001.04536.

15 *Markidis, Stefano.* The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers? / Stefano Markidis // *Frontiers in Big Data.* — 2021. — Vol. 4. https://www.frontiersin.org/article/10.3389/fdata.2021.669097.

16 *Han, Jiequn.* Solving high-dimensional partial differential equations using deep learning / Jiequn Han, Arnulf Jentzen, Weinan E // *Proceedings of the National Academy of Sciences.* — 2018. — Vol. 115, no. 34. — Pp. 8505–8510. https://www.pnas.org/doi/abs/10.1073/pnas.1718942115.

17 Characterizing possible failure modes in physics-informed neural networks / Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe et al. — 2021. https://arxiv.org/abs/2109.01050.

18 Google Colab. https://colab.research.google.com/.

19 *Colton, David.* The Numerical Solution of the Inverse Stefan Problem in Two Space Variables / David Colton, Rembert Reemtsen // *SIAM Journal on Applied Mathematics.* — 1984. — Vol. 44, no. 5. — Pp. 996–1013. http://www.jstor.org/stable/2101132.

20 *Johansson, B. Tomas.* The method of fundamental solutions for the two-dimensional inverse Stefan problem / B. Tomas Johansson, Daniel Lesnic, Thomas Reeve // *Inverse Problems in Science and Engineering.* — 2014. — Vol. 22, no. 1. — Pp. 112–129. https://doi.org/10.1080/17415977.2013.827180.

21 *Tomas Johansson, B.* A Meshless Regularization Method for a Two-Dimensional Two-Phase Linear Inverse Stefan Problem / B. Tomas Johansson, Daniel Lesnic, Thomas Reeve // *Advances in Applied Mathematics and Mechanics.* — 2013. — Vol. 5, no. 6. — P. 825–845.

22 *Wang, Xin.* A Survey on Curriculum Learning. — 2020. https://arxiv.org/abs/2010.13166.