

UNIVERSITY OF MICHIGAN-FLINT

MASTER'S THESIS

**Applying Physics-Informed Neural Networks to
the Solution of Parametric, Nonlinear Heat
Conduction Problems**

Author:
Collin RUSSELL

Supervisor:
Dr. Matthew SPRADLING

*A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science*


in

Computer Science & Information Systems

of the

College of Innovation & Technology

December 18, 2023

1st 
Reader: _____
Dr. Matthew SPRADLING

2nd 
Reader: _____
Dr. James ALSUP

Declaration of Authorship

I, Collin RUSSELL, declare that this thesis titled, “Applying Physics-Informed Neural Networks to the Solution of Parametric, Nonlinear Heat Conduction Problems” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: *Collin Russell*

Date: *12/18/2023*

UNIVERSITY OF MICHIGAN-FLINT

Abstract

College of Innovation & Technology

Master of Science

Applying Physics-Informed Neural Networks to the Solution of Parametric, Nonlinear Heat Conduction Problems

by Collin RUSSELL

Physics-informed neural networks (PINNs), and the many variants they have inspired, have the potential to reshape how scientists and engineers solve computational physics problems. Despite their relatively recent introduction to the scientific community, PINNs have been successfully used to approximate the solution to a variety of computational physics problems, including heat transfer problems. In this work, a PINN is trained to solve a 2-D, steady-state heat conduction problem inspired by the practical problem of finding the cross-sectional, steady-state temperature field in the insulation module of a box furnace. The problem is formulated such that (i) the thermal conductivity of the heat-conducting medium is modeled as quadratic function of temperature and (ii) the coefficients of the quadratic function are parameterized and passed to the PINN as additional inputs. The PINN is trained exclusively by minimizing physics residuals (i.e., no data is used in the training process), and its accuracy is evaluated using a testing dataset composed of eight numerical solutions, each of which is associated with a unique, commercially available high-temperature insulation material. The results obtained in this work demonstrate the feasibility of training a PINN (without any training data) to accurately solve heat conduction problems involving parameterized, temperature-dependent material property models. To the author's knowledge, this is the first time this capability has been demonstrated.

Acknowledgements

I would like to thank Dr. Spradling for his guidance over the course of completing this thesis. His propensity to ask the right questions and his ability to identify weaknesses in logical arguments were particularly appreciated, in addition to his positive, encouraging attitude. I would also like to thank my wife and my son for their unwavering love, support, and patience over the past year; I am immensely grateful to play this game of life on such a wonderful team.

Contents

Declaration of Authorship	iii
Abstract	v
Acknowledgements	vii
1 Introduction	1
2 Literature Review	3
3 Theoretical Background	7
3.1 Feedforward Neural Networks	7
3.2 Physics-Informed Neural Networks	9
4 Problem Description	13
4.1 Introduction to the Problem	13
4.2 Dimensional Form of the Problem	14
4.3 Dimensionless Form of the Problem	19
5 Methodology & Implementation	21
5.1 Generation of Testing Data	21
5.2 Architecture of the PINN	22
5.3 Training the PINN	25
6 Results	31
6.1 Presentation of Results	31
6.2 Discussion of Results	34
7 Conclusion	37
7.1 Contributions of this Work	37
7.2 Opportunities for Future Research	37
8 Appendix I	39
9 Appendix II	43
10 Appendix III	45

List of Figures

3.1	A three-layer FNN composed of a single-node input layer (gray), a three-node hidden layer (blue), a two-node hidden layer (blue), and a single-node output layer (green).	8
3.2	A prototypical 2-D, steady-state, linear heat conduction problem.	11
3.3	Schematic of the PINN used to approximate the solution to the example problem shown in Fig. 3.2.	12
4.1	A resistively heated, benchtop muffle furnace [27].	13
4.2	The general, dimensional form of the heat conduction problem considered in this work.	14
4.3	The nonlinear, dimensional form of the problem.	16
4.4	Tabulated temperature-dependent thermal conductivity data and corresponding best-fit quadratic equations for the eight reference insulation materials.	18
4.5	Computed quadratic equation coefficients and corresponding coefficients of determination for the eight reference insulation materials.	18
4.6	The nonlinear, dimensionless form of the problem.	20
5.1	Numerical solution $u^*(x^*, y^*)$ for each problem instance, i.e., each reference material, included in the testing dataset.	23
5.2	Domain collocation points projected onto 2-D plane defined by x^*, y^* .	24
5.3	Schematic of the PINN used to approximate the solution to the problem of interest.	26
5.4	Domain collocation points projected onto the 3-D spaces defined by x^*, y^*, α (top), x^*, y^*, β (middle), and x^*, y^*, η (bottom).	27
5.5	Boundary collocation points projected onto the 3-D spaces defined by x^*, y^*, α (top), x^*, y^*, β (middle), and x^*, y^*, η (bottom).	28
5.6	Thermal conductivity vs. temperature curves associated with collocation points generated by uniformly sampling the dimensionless parameter space specified in Table 4.3. The number of training curves (black) included in the top plot is 800, the minimum number of collocation points assigned to any subdomain/boundary, and the number included in the bottom plot is 12000, the maximum number of collocation points assigned to any subdomain/boundary.	29
6.1	Solution prediction $\hat{u}^*(x^*, y^*)$ for each problem instance, i.e., each reference material, included in the testing dataset.	32

6.2	Error $\hat{u}^*(x^*, y^*) - u^*(x^*, y^*)$ for each problem instance, i.e., each reference material, included in the testing dataset.	33
6.3	Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 1. A key for the boundary numbering scheme used is provided in Fig. 6.4.	35
6.4	Key for the boundary numbering scheme used in Fig. 6.3.	36
8.1	Estimated natural convection HTC's along each physically unique surface of the furnace insulation module, assuming $u_s = 440^\circ\text{K}$ and thus $u_s^* = 0.346$	40
8.2	Estimated natural convection HTC's along each physically unique surface of the furnace insulation module, assuming $u_s = 610^\circ\text{K}$ and thus $u_s^* = 0.479$	41
10.1	Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 2.	46
10.2	Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 3.	47
10.3	Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 4.	48
10.4	Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 5.	49
10.5	Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 6.	50
10.6	Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 7.	51
10.7	Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 8.	52

List of Tables

4.1	The dimensional parameter space.	17
4.2	Product name and manufacturer for each of the eight reference insulation materials [28]–[32].	17
4.3	The dimensionless parameter space.	20
5.1	The weights used to compute the PINN’s composite loss.	30
6.1	RRMSE of each solution prediction.	34

1 Introduction

Computational physics problems generally involve solving differential equations. Numerical methods such as the finite difference method (FDM), finite volume method (FVM), and finite element method (FEM) are used extensively in this discipline because most real-world problems involve complex geometries, mixed boundary conditions, and/or nonlinearities that preclude the use of analytical solution methods [1], [2]. Numerical methods, however, are not without disadvantages; applying numerical methods to the solution of computational physics problems typically requires:

1. Discretizing the domain of interest;
2. Transforming the governing differential equation(s) and imposed boundary and/or initial condition(s) into approximate, algebraic equations valid at discrete points in space and/or time; and
3. Solving the resultant system(s) of equations to obtain an approximate solution, which can be a computationally expensive, time-intensive process.

Data-driven models leveraging the function approximation capabilities of neural networks [3], [4] are an attractive alternative to models applying numerical methods because they are not subject to the requirements above and can, once trained, return accurate results quickly with minimal computational effort. The availability of graphics processing units (GPUs) capable of massively parallel, general-purpose computation (e.g., NVIDIA's CUDA-compatible GPUs [5]) and the accessibility of powerful, open-source machine learning software (e.g., Python's PyTorch [6] and TensorFlow [7] libraries) also contribute to the increasing interest in applying neural networks to computational physics problems. Conventional (i.e., physics-naive) neural networks, however, typically require large training datasets to accurately approximate a function over the domain of interest. When applied to problems for which data acquisition, including the collection of empirically derived ground-truth data and/or the generation of simulation-derived data, is prohibitively expensive or time intensive, this reliance on training data can be a major limitation. Many computational physics problems, including heat transfer problems, fall into this so-called "small data" regime [8], which diminishes the utility of conventional neural networks in this application space.

Physics-informed neural networks (PINNs) cleverly overcome the data dependence of conventional neural networks by enabling the incorporation of prior knowledge, often laws of physics expressed in the form of differential equations, into their training. More specifically,

PINNs replace (or augment) the computation of data-based loss with the computation of physics-based loss that indicates how closely the network's predictions abide by the governing differential equation(s) and imposed boundary and/or initial condition(s) [8]. By minimizing this loss, PINNs are trained to respect the laws of physics. This incorporation of prior knowledge – information that conventional neural networks cannot utilize – enables PINNs to solve complex physics problems with a relatively high degree of accuracy, even when applied to problems for which little (or zero) training data is available [8].

The literature contains numerous examples of applying PINNs to the solution of computational physics problems, including computational heat transfer problems. Similarly, the literature contains multiple examples of training PINNs without training data. This work is unique in that it applies a PINN to the solution of a nonlinear heat transfer problem, more specifically a nonlinear heat conduction problem, in which material properties are assumed to be temperature dependent and are modeled using parametric functions of temperature. Training a PINN to accurately solve this type of problem over a reasonably large parameter space would demonstrate that a single PINN can be trained to solve a general heat conduction problem for many different nonlinear materials.¹ To the author's knowledge, this has never been shown previously.

The remainder of this paper is organized as follows:

- Chapter 2 reviews related works in the literature;
- Chapter 3 presents the fundamental theoretical aspects of PINNs in the context of feedforward neural networks, which provide the foundation upon which most PINNs are built;
- Chapter 4 specifies the heat conduction problem considered in this work;
- Chapter 5 describes the PINN configuration applied in this work and the methods employed to generate the testing data used to evaluate the PINN's accuracy;
- Chapter 6 presents the PINN's post-training solution predictions and discusses the magnitude and likely causes of the observed prediction error; and
- Chapter 7 summarizes the value of this work and identifies opportunities for future research.

¹Here, the term "nonlinear materials" refers to materials with temperature-dependent properties. This temperature dependency makes the governing differential equation (the heat equation) nonlinear.

2 Literature Review

The theoretical capability of neural networks to approximate the solutions to differential equations has been established for decades. In proving that multilayer feedforward neural networks (FNNs) are universal function approximators, Hornik et al. [3] and Cybenko [4] implicitly proved that FNNs of sufficient size and with sufficient training can accurately approximate the solutions to differential equations, including the differential equations that govern heat transfer phenomena.

The application of neural networks to the solution of heat transfer problems is also far from new. In 1989, the same year that Hornik et al. and Cybenko's works were published, Watanabe et al. used a multilayer FNN to detect incipient thermal-, chemical-, and flow-related faults in a temperature-controlled chemical reactor [9]. Shortly thereafter, Thibault et al. applied FNNs to a set of data correlation tasks including mapping the voltage generated by a type-K thermocouple to the thermocouple's temperature and mapping the Rayleigh number to the Nusselt number in problems involving natural convection heat transfer along horizontal cylinders [10].¹

Numerous examples of using neural networks to solve heat conduction problems emerged in the following years:

- Dissanayake et al. used FNNs to predict the steady-state temperature field in a 2-D region subject to temperature-dependent internal heat generation and mixed boundary conditions [11].
- Jambunathan et al., in one of the earliest examples of applying neural networks to the solution of inverse heat conduction problems, used FNNs to predict the time-dependent, spatially averaged convection heat transfer coefficient along the external surface of a semi-infinite wall heated to an elevated initial temperature [12].
- Diaz et al. used neural networks to solve a set of heat transfer problems of increasing complexity; the authors first demonstrated how a FNN with as few as one hidden node can accurately predict the steady-state conduction heat transfer rate through a wall, and ultimately used a multilayer FNN to predict the steady-state rate of heat transfer in a finned-tube heat exchanger [13].

The preceding application examples illustrate the merit of using neural networks to solve computational heat transfer problems. It is important to note, however, that in all of these

¹The Rayleigh number is dimensionless parameter indicating the relative magnitude of buoyancy forces vs. viscous forces in a flow, and the Nusselt number is a dimensionless parameter indicating the magnitude of the temperature gradient at a fluid-surface-fluid boundary [1], [2].

examples (i) the accuracy of the neural networks was reliant upon the availability of data derived from known analytical solutions, approximate numerical solutions, or laboratory experiments; and (ii) the predictions output by the neural networks, while generally accurate (due to the availability of sufficient training data), were in no way constrained by underlying laws of physics. It is these limitations, which are inherent to conventional neural networks, that PINNs address.

Raissi et al. introduced PINNs in 2019;² however, the works of Psychogios et al. [16] and Lagaris et al. [17] inspired and provided much of the theoretical foundation for their development of the PINN framework [8], [18]. Psychogios et al. demonstrated that “hybrid” process models composed of a neural network and a first-principles physics model incorporating prior knowledge can significantly increase the accuracy and reduce the training data requirements of process models composed of a neural network alone [16]. A few years later, Lagaris et al. presented a novel method of training neural networks to approximate the solution to differential equations that eliminates the need for data-based training. Given a general differential equation of the form $f(\mathbf{x}) = g(\mathbf{x}, y(\mathbf{x}), \nabla y(\mathbf{x}), \nabla^2 y(\mathbf{x}), \dots)$, $\mathbf{x} \in \Omega$ with an unknown particular solution $y(\mathbf{x})$ associated with a given set of boundary and/or initial conditions, their method consists of [17]:

1. Formulating a trial solution of the form $\hat{y}(\mathbf{x}) = a(\mathbf{x}) + b(\mathbf{x}, z(\mathbf{x}; \theta))$, where a and b are functions dependent upon the form of the differential equation g , and $z(\mathbf{x}; \theta)$ is the output of a FNN $f_{NN}(\mathbf{x}; \theta)$ with parameters θ ;
2. Selecting a set of N collocation points $\{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N\}$ in the domain Ω ;
3. Training the neural network $f_{NN}(\mathbf{x}; \theta)$ with the objective of minimizing $\mathcal{L} = \sum_{i=1}^N \left(g(\mathbf{x}_i, \hat{y}(\mathbf{x}_i), \nabla \hat{y}(\mathbf{x}_i), \nabla^2 \hat{y}(\mathbf{x}_i), \dots) \right)^2$, the mean-squared error (MSE) of the trial solution evaluated over the collocation points $\{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N\}$; and
4. Returning the trial solution $\hat{y}(\mathbf{x})$ as the resultant approximation of $y(\mathbf{x})$.

The PINN framework presented by Raissi et al. is essentially a modern variant of this method that uses *automatic differentiation* (instead of manually derived derivative expressions) to compute the derivatives $\frac{\partial \hat{y}}{\partial x_i}$ needed to compute the loss \mathcal{L} .³

Despite their relative youth, PINNs have been applied to the solution of many computational heat transfer problems. Some of the most prominent examples are presented in Cai et al.’s and Hennigh et al.’s works [18], [19]. Cai et al. applied PINNs to the prediction of 2-D fluid pressure, velocity, and temperature fields in steady-state and transient convection heat transfer problems. The authors also applied PINNs to an inverse, transient heat conduction problem involving a dynamic phase-transformation boundary in a prototypical 2-D domain [18]. Hennigh et al. presented NVIDIA SimNet, a TensorFlow-based framework for solving parametric,

²Raissi et al. posted preprints [14], [15] of their seminal paper in 2017.

³PINNs also use automatic differentiation to compute derivatives of the form $\frac{\partial \mathcal{L}}{\partial \theta}$, which are needed to optimize the network parameters θ [8].

complex-geometry, multiphysics problems using PINNs and demonstrated its capabilities using a number of test cases including (i) predicting the 3-D, steady-state pressure, velocity, and temperature fields in/around a finned FPGA heat sink contained in a rectangular channel and (ii) optimizing the parameterized geometry of a heat sink using a 3-D conjugate heat transfer model [19].

The literature also contains numerous works that apply PINNs to problems primarily or exclusively involving conduction heat transfer (i.e., problems primarily or exclusively involving the solution of the heat equation). For example:

- Zobeiry et al. applied PINNs to the solution of a prototypical 2-D, transient heat conduction problem involving parameterized convection heat transfer boundary conditions and trained the PINN using an exclusively physics-based, i.e., data-free, training process [20].
- Liao et al. used a PINN to predict the 3-D, transient temperature field in rectangular slab subject to localized heat generation from a moving heat source, simulating heat transfer during a direct-energy-deposition metal additive manufacturing process. The authors investigated the use of both data-reliant and data-free training methods [21].
- Wurth et al. applied PINNs to predict the 2-D, transient temperature field in a rectangular domain undergoing an exothermic thermochemical curing process. The authors parameterized the boundary conditions and material properties to capture the effects of varying process parameters (e.g. the setpoint autoclave temperature) and material compositions. They also used an exclusively physics-based training process [22].

This thesis is differentiated from these earlier works in its application of PINNs to the solution of conduction heat transfer problems in which material properties are assumed to be temperature-dependent and modeled as parametric functions of temperature. While PINNs have previously been applied to the solution of parametric and/or nonlinear heat conduction problems (e.g., Zobeiry et al. considered a problem involving parameterized boundary conditions [20], Liao et al. considered a nonlinear problem involving temperature dependent properties [21], and Wurth et al. considered a problem involving parameterized, temperature-independent material properties [22]), the application of PINNs to heat conduction problems involving parameterized, temperature-dependent material property models has, to the author's knowledge, never been investigated.

3 Theoretical Background

This chapter presents the theoretical basis of PINNs and demonstrates how they are applied using a simple example problem. Because PINNs are prototypically built upon FNNs [8], the chapter begins with a brief review of the function, structure, and mathematical representation of FNNs.

3.1 Feedforward Neural Networks

Functionally, FNNs are function approximators. A FNN approximating $\mathbf{y} = f(\mathbf{x})$ outputs $\hat{\mathbf{y}} = f_{NN}(\mathbf{x}; \boldsymbol{\theta})$, where the network's parameters $\boldsymbol{\theta}$ (weights and biases) are optimized during training to minimize the error in each prediction $\hat{\mathbf{y}}^{[k]} = f_{NN}(\mathbf{x}^{[k]}; \boldsymbol{\theta})$ relative to the corresponding true value $\mathbf{y}^{[k]} = f(\mathbf{x}^{[k]})$ [23].

Structurally, a FNN is a collection of nodes, a.k.a. artificial neurons, arranged in sequentially interconnected layers. This structure is illustrated in Fig. 3.1 for a small three-layer network.¹ Each node in a FNN computes a biased, weighted sum of its inputs, passes the sum to an activation function, and outputs the value returned. Due to how nodes are interconnected in FNNs, the output of each computational layer is:

$$\mathbf{a}^{(i)} = \sigma^{(i)}(\mathbf{W}^{(i)}\mathbf{x}^i + \mathbf{b}^{(i)}), \quad (3.1)$$

where $\mathbf{a}^{(i)}$ contains the output of each node in layer i ; $\sigma^{(i)}$ is the (typically nonlinear) activation function applied by the nodes in layer i ; $\mathbf{W}^{(i)}$ contains the weights associated with each node in layer i ; $\mathbf{x}^{(i)}$ contains the inputs to layer i and is equal to $\mathbf{a}^{(i-1)}$, the output of the preceding layer; and $\mathbf{b}^{(i)}$ contains the bias of each node in layer i [23].² Using the FNN shown in Fig. 3.1 as an example, the output of each computational layer in the network is:

¹The FNN shown in Fig. 3.1 appears to have three layers, however the input (first) layer is typically excluded from layer count because it is not a computational layer.

²This mathematical representation implicitly assumes the FNN is fully connected. In this work, all of the FNNs presented are fully connected FNNs.

$$\begin{aligned}
\mathbf{a}^{(1)} &= \sigma^{(1)}(\mathbf{w}^{(1)}\mathbf{a}^{(0)} + \mathbf{b}^{(1)}) = \sigma^{(1)}\left(\begin{bmatrix} w_{0\rightarrow0}^{(1)} \\ w_{0\rightarrow1}^{(1)} \\ w_{0\rightarrow2}^{(1)} \end{bmatrix} a_0^{(0)} + \begin{bmatrix} b_0^{(1)} \\ b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}\right) \\
\mathbf{a}^{(2)} &= \sigma^{(2)}(\mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)}) = \sigma^{(2)}\left(\begin{bmatrix} w_{0\rightarrow0}^{(2)} & w_{1\rightarrow0}^{(2)} & w_{2\rightarrow0}^{(2)} \\ w_{0\rightarrow1}^{(2)} & w_{1\rightarrow1}^{(2)} & w_{2\rightarrow1}^{(2)} \end{bmatrix} \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \end{bmatrix} + \begin{bmatrix} b_0^{(2)} \\ b_1^{(2)} \end{bmatrix}\right) \quad (3.2) \\
\hat{y} = a^{(3)} &= \sigma^{(3)}(\mathbf{w}^{(3)}\mathbf{a}^{(2)} + b^{(3)}) = \sigma^{(3)}\left(\begin{bmatrix} w_{0\rightarrow0}^{(3)} & w_{1\rightarrow0}^{(3)} \end{bmatrix} \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \end{bmatrix} + b_0^{(3)}\right).
\end{aligned}$$

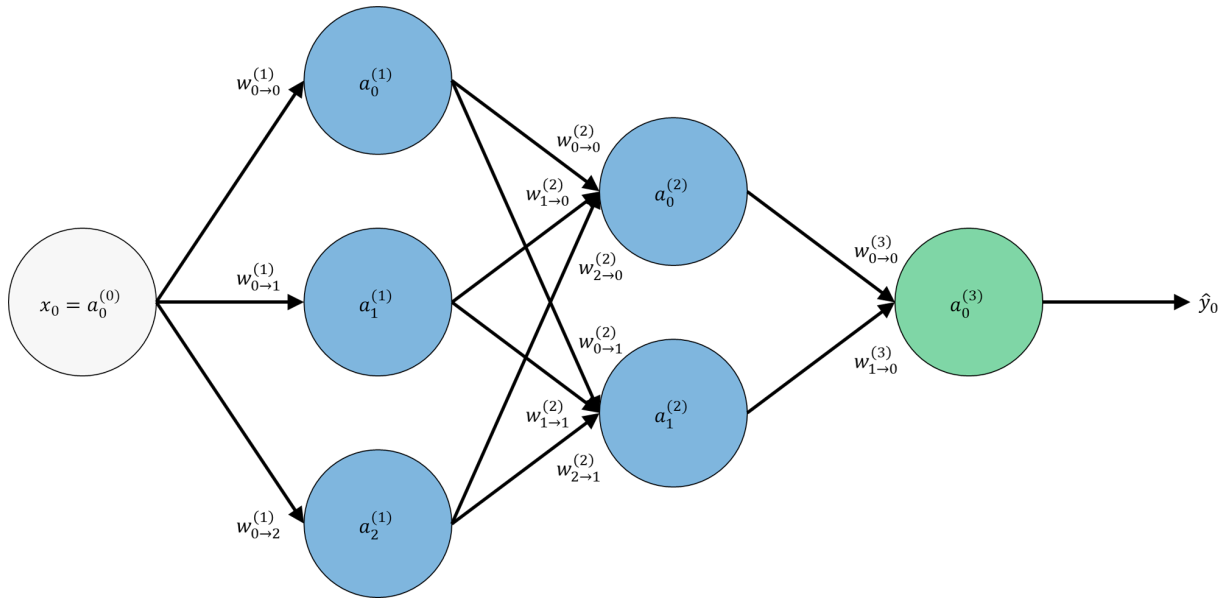


FIGURE 3.1: A three-layer FNN composed of a single-node input layer (gray), a three-node hidden layer (blue), a two-node hidden layer (blue), and a single-node output layer (green).

A FNN $f_{NN}(\mathbf{x}; \boldsymbol{\theta})$ can therefore be represented as a sequence of function compositions in which the number of composed functions is equal to L , the number of computational layers in the network. Accordingly:

$$f_{NN}(\mathbf{x}; \boldsymbol{\theta}) = f_{NN}^{(L)}\left(f_{NN}^{(L-1)}\left(\dots\left(f_{NN}^{(2)}\left(f_{NN}^{(1)}(\mathbf{x}; \boldsymbol{\theta}^{(1)}); \boldsymbol{\theta}^{(2)}\right)\dots\right); \boldsymbol{\theta}^{(L-1)}\right); \boldsymbol{\theta}^{(L)}\right), \quad (3.3)$$

where, from Eq. 3.1, each composed function is of the form $f_{NN}^{(i)} = \sigma^{(i)}(\mathbf{W}^{(i)}\mathbf{x}^i + \mathbf{b}^{(i)})$ [23]. Note again that $\mathbf{x}^i = \mathbf{a}^{(i-1)} = f_{NN}^{(i-1)}(\mathbf{x}^{(i-1)}; \boldsymbol{\theta}^{(i-1)})$ [23].

FNNs are trained (and tested) prior to their deployment. Training a FNN is an iterative process consisting of *forward propagation*, *error computation*, *backward propagation*, and *parameter optimization*.

- Forward propagation is the propagation of each input $\mathbf{x}^{[k]}$ through the network to generate a prediction $\hat{\mathbf{y}}^{[k]} = f_{NN}(\mathbf{x}^{[k]}; \boldsymbol{\theta})$, where $\mathbf{x}^{[k]}$ contains the inputs from a single training example k and $\hat{\mathbf{y}}^{[k]}$ contains the corresponding predictions.
- Error computation consists of passing each prediction $\hat{\mathbf{y}}^{[k]}$ and its corresponding true value $\mathbf{y}^{[k]}$ to a loss function, a.k.a. cost function, to compute a scalar loss \mathcal{L} representative of the aggregate error in the network's predictions [23].
- Backward propagation, a.k.a. *backpropagation*, is the propagation of loss \mathcal{L} back through the network to determine the impact of each network parameter θ on the loss, where the impact is expressed as a partial derivative of the form $\frac{\partial \mathcal{L}}{\partial \theta}$ [23]. In modern FNNs, these partial derivatives are typically computed by *reverse-mode automatic differentiation* (hereafter referred to simply as automatic differentiation), a differentiation technique that applies the chain rule of calculus and dynamic programming to compute derivatives at machine precision (i.e., practically exactly) with high computational efficiency [24].
- Parameter optimization refers to the computation of new network parameters with the objective of minimizing the network's loss \mathcal{L} . New parameter values are often computed using a gradient-based optimization algorithm, which takes the partial derivatives computed during backward propagation as inputs [23].

3.2 Physics-Informed Neural Networks

PINNs are differentiated from FNNs in how they are trained. In the context of solving forward, a.k.a. direct, physics problems governed by differential equations, a FNN is trained with the objective of minimizing the error in its predictions relative to a corresponding set of true values (where the predictions collectively comprise the predicted solution of the problem and the true values collectively comprise the true solution of the problem). In contrast, a PINN is trained with the objective of minimizing the residuals of the differential equation(s) and associated boundary and/or initial conditions applicable to the problem [8]. Accordingly, while FNNs can only be trained to respect the laws of physics relevant to a given problem *indirectly* (i.e., without any constraints imposed by the differential equation(s) and associated boundary and/or initial conditions that express these laws of physics), PINNs are trained to respect the relevant laws of physics *directly*.

Applied to a general heat transfer problem governed by the source-free heat equation, a PINN approximates the solution to a differential equation of the form [8]:

$$\begin{aligned}
 \rho c \frac{\partial u}{\partial t} &= \frac{\partial}{\partial x} \left(k \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial u}{\partial y} \right) + \frac{\partial}{\partial z} \left(k \frac{\partial u}{\partial z} \right), & x, y, z \in \Omega, & t \in [0, T] \\
 u(x, y, z, t) &= g(x, y, z, t), & x, y, z \in \partial\Omega, & t \in [0, T] \\
 u(x, y, z, 0) &= h(x, y, z), & x, y, z \in \Omega, &
 \end{aligned} \tag{3.4}$$

where x , y , and z are the spatial coordinates; t is the temporal coordinate; ρ , c , and k are the density, specific heat capacity, and thermal conductivity of the material(s) in the domain Ω ; and $u(x, y, z, t)$ is the unknown solution of the differential equation with boundary conditions $g(x, y, z, t)$ and initial condition $h(x, y, z)$. This is accomplished by creating a FNN to approximate $u(x, y, z, t)$ and optimizing its parameters to minimize a composite, weighted loss function of the form:

$$\mathcal{L} = \lambda_D \mathcal{L}_D + \lambda_B \mathcal{L}_B + \lambda_I \mathcal{L}_I, \quad (3.5)$$

where \mathcal{L} is the composite loss; \mathcal{L}_D , \mathcal{L}_B , and \mathcal{L}_I are physics-based loss components associated with the differential equation, boundary conditions, and initial condition respectively; and λ_D , λ_B , and λ_I are weights corresponding to loss components \mathcal{L}_D , \mathcal{L}_B , and \mathcal{L}_I respectively [8], [18].³

If the MSE loss function is applied, as it is in this work, \mathcal{L}_D , \mathcal{L}_B , and \mathcal{L}_I are:

$$\begin{aligned} \mathcal{L}_D &= \frac{1}{N_D} \sum_{k=1}^{N_D} \left(\mathcal{R}_D^{[k]} \right)^2 \\ \mathcal{L}_B &= \frac{1}{N_B} \sum_{k=1}^{N_B} \left(\mathcal{R}_B^{[k]} \right)^2 \\ \mathcal{L}_I &= \frac{1}{N_I} \sum_{k=1}^{N_I} \left(\mathcal{R}_I^{[k]} \right)^2, \end{aligned} \quad (3.6)$$

where the differential equation, boundary condition, and initial condition residuals $\mathcal{R}_D^{[k]}$, $\mathcal{R}_B^{[k]}$, and $\mathcal{R}_I^{[k]}$ associated with a given collocation point k , are [8]:

$$\begin{aligned} \mathcal{R}_D^{[k]} &= \rho c \frac{\partial \hat{u}^{[k]}}{\partial t^{[k]}} - \frac{\partial}{\partial x^{[k]}} \left(k \frac{\partial \hat{u}^{[k]}}{\partial x^{[k]}} \right) - \frac{\partial}{\partial y^{[k]}} \left(k \frac{\partial \hat{u}^{[k]}}{\partial y^{[k]}} \right) - \frac{\partial}{\partial z^{[k]}} \left(k \frac{\partial \hat{u}^{[k]}}{\partial z^{[k]}} \right) \\ \mathcal{R}_B^{[k]} &= g(x^{[k]}, y^{[k]}, z^{[k]}, t^{[k]}) \\ \mathcal{R}_I^{[k]} &= h(x^{[k]}, y^{[k]}, z^{[k]}); \end{aligned} \quad (3.7)$$

and N_D , N_B , and N_I are the number of collocation points used to enforce the differential equation, boundary conditions, and initial condition respectively during the training process.

It should be noted that PINNs can also be trained with data, in which case an additional data-based loss component \mathcal{L}_{data} and a corresponding weight λ_{data} are added to Eq. 3.5 [8]. In such cases, the physics-based, *unsupervised learning* process is combined with a data-based, *supervised learning* process.

As a simple example, consider the linear, nonparametric heat conduction problem shown schematically in Fig. 3.2. The objective of the problem is to find the steady-state temperature field $u(x, y)$ in the domain given the imposed temperature (Dirichlet) boundary conditions applied to its left and bottom boundaries and the convection (Robin) boundary conditions applied to its right and top boundaries. This problem is of the form:

³The composite loss function shown in Eq. 3.5 implicitly assumes that the boundary conditions are enforced as “soft” constraints (i.e., via loss penalization). Alternative approaches enable enforcing the boundary conditions as “hard” constraints, which force the predictions to satisfy the boundary conditions automatically (i.e., without loss penalization) [17], [25], [26].

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= 0, & x \in (0, L), y \in (0, L) \\
u &= u_0, & x \in [0, L], y = 0 \\
u &= u_0, & x = 0, y \in [0, L] \\
-k \frac{\partial u}{\partial y} &= h(u - u_\infty), & x \in (0, L), y = L \\
-k \frac{\partial u}{\partial x} &= h(u - u_\infty), & x = L, y \in (0, L),
\end{aligned} \tag{3.8}$$

where x and y are the spatial coordinates, both of which are bounded by $[0, L]$; u_0 is the imposed temperature along the left and bottom boundaries; u_∞ is the ambient temperature of the surrounding fluid; k is the thermal conductivity of the domain; h is the convection heat transfer coefficient (HTC) along the right and top boundaries; and $u(x, y)$ is the unknown solution of the differential equation with the specified boundary conditions.

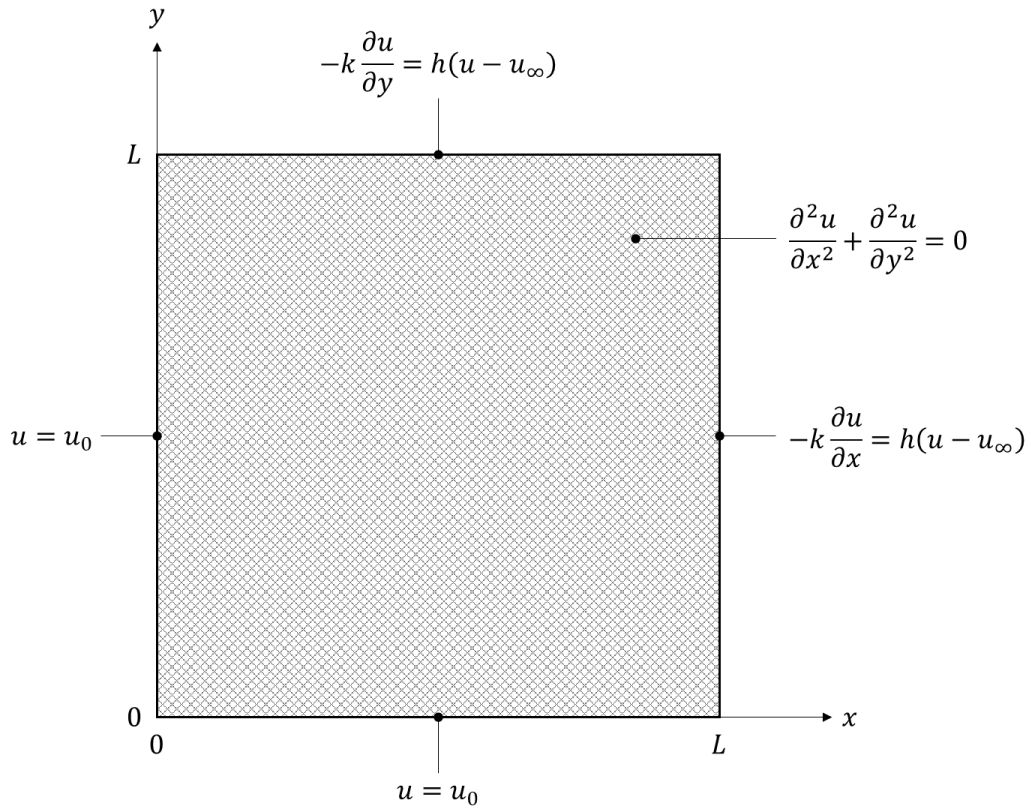


FIGURE 3.2: A prototypical 2-D, steady-state, linear heat conduction problem.

The solution to this problem, the steady-state temperature field $u(x, y)$, can be approximated by the PINN shown in Fig. 3. This PINN is built upon a two-input, one-output FFN composed of L computational layers and M nodes per hidden layer. Given a pair of inputs x, y , the FNN outputs the corresponding prediction $\hat{u}(x, y)$, i.e., the predicted temperature at position x, y .

During training, the physics module applies automatic differentiation to differentiate each

prediction $u^{[k]}$ with respect to $x^{[k]}$ and $y^{[k]}$ and uses these derivatives to compute the physics-based loss components \mathcal{L}_D and \mathcal{L}_B . The composite loss \mathcal{L} is then differentiated with respect to the network parameters θ and the resultant derivatives, also computed using automatic differentiation, are used to optimize the network parameters such that \mathcal{L} is minimized.

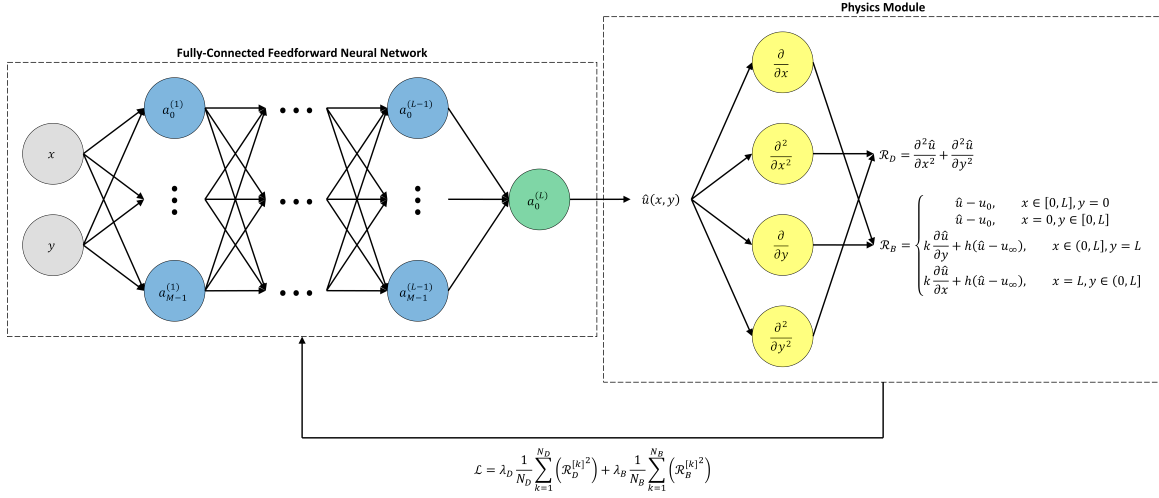


FIGURE 3.3: Schematic of the PINN used to approximate the solution to the example problem shown in Fig. 3.2.

4 Problem Description

This chapter specifies the problem considered in this work, hereafter referred to as “the heat conduction problem” or simply “the problem”. From the problem statement, a dimensional form of the problem is formulated, from which an equivalent dimensionless form is derived.

4.1 Introduction to the Problem

In this work, a PINN is used to approximate the solution to 2-D, steady-state heat conduction problem inspired by, but simplified from, the task of finding the cross-sectional, steady-state temperature field in the insulation module of a box furnace such as the muffle furnace shown in Fig. 4.1. Knowing the approximate steady-state temperature field in the insulation module of such a furnace is of practical value to furnace designers because it can be used to estimate the furnace’s steady-state rate of heat loss and can also inform the insulation selection process.

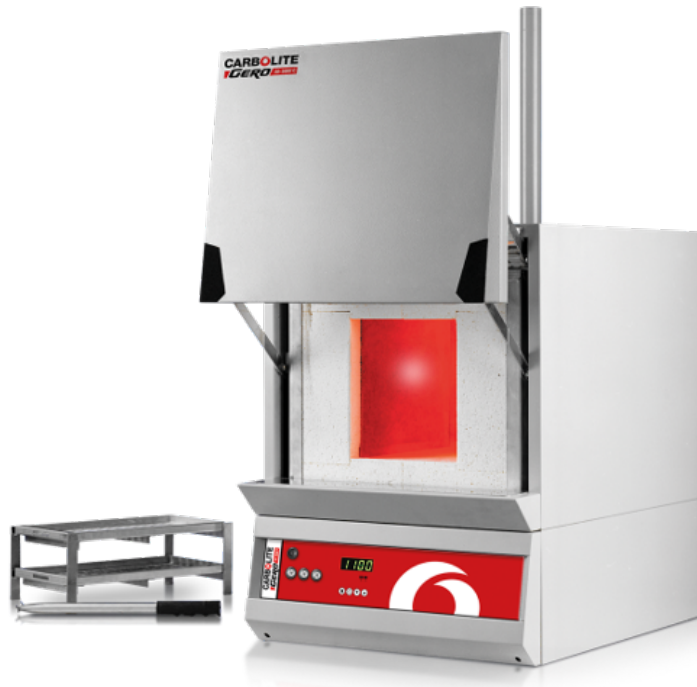


FIGURE 4.1: A resistively heated, benchtop muffle furnace [27].

4.2 Dimensional Form of the Problem

This heat conduction problem, shown schematically in Fig. 4.2, is of the general form:

$$\begin{aligned}
 \frac{\partial}{\partial x} \left(k \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial u}{\partial y} \right) &= 0, & x \in (L_{int}, L_{ext}), y \in (0, L_{int}] \\
 & & x \in (L_{int}, L_{ext}), y \in (L_{int}, L_{ext}) \\
 & & x \in (0, L_{int}), y \in (L_{int}, L_{ext}) \\
 \frac{\partial u}{\partial y} &= 0, & x \in (L_{int}, L_{ext}), y = 0 \\
 u &= u_0, & x = L_{int}, y \in [0, L_{int}] \\
 u &= u_0, & x \in [0, L_{int}], y = L_{int} \\
 \frac{\partial u}{\partial x} &= 0, & x = 0, y \in (L_{int}, L_{ext}) \\
 -k \frac{\partial u}{\partial y} &= h(u - u_\infty), & x \in [0, L_{ext}], y = L_{ext} \\
 -k \frac{\partial u}{\partial x} &= h(u - u_\infty), & x = L_{ext}, y \in [0, L_{ext}],
 \end{aligned} \tag{4.1}$$

where x and y are the spatial coordinates; L_{int} and L_{ext} are the lengths that define the geometry of the domain; u_0 is the imposed internal temperature; u_∞ is the ambient air temperature; k is the thermal conductivity of the domain; h is the convection HTC along the boundaries subject to natural convection heat transfer; and $u(x, y)$ is the unknown solution of the differential equation with the specified boundary conditions.

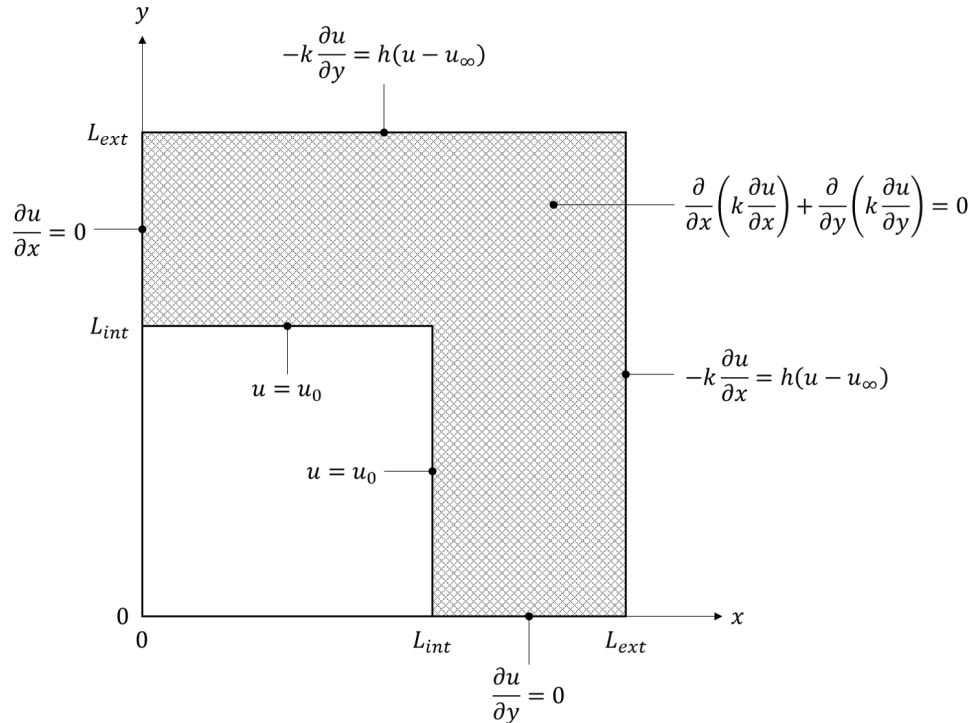


FIGURE 4.2: The general, dimensional form of the heat conduction problem considered in this work.

This problem involves mixed boundary conditions, as detailed below:

- Imposed temperature (Dirichlet) boundary conditions are applied to the internal boundaries of the domain, the boundaries representative of the internal surfaces of the furnace insulation module. These boundary conditions impose a uniform temperature to these surfaces that is assumed to be equal to the setpoint temperature of the furnace chamber.
- Imposed heat flux (Neumann) boundary conditions are applied to the domain boundaries coincident with the x- and y-axes. By imposing a heat flux of zero, these boundary conditions imply symmetry about the x- and y-axes.
- Convection (Robin) boundary conditions are applied to the external boundaries of the domain, the boundaries representative of the external surfaces of the furnace insulation module. These boundary conditions account for natural convection heat transfer along these surfaces.

Additionally, because the insulation materials utilized in high-temperature furnaces often have highly temperature-dependent thermal conductivities, the thermal conductivity of the domain is assumed to be a function of temperature. Based upon a review of temperature-dependent thermal conductivity data in the literature [28]–[32], the thermal conductivity of the domain was modeled using a quadratic equation of the form:

$$k(u) = au^2 + bu + c \quad (4.2)$$

where the coefficients a , b , and c are unique to each insulation material and can be computed via quadratic regression given tabulated thermal conductivity vs. temperature data. The temperature dependency of k described in Eq. 4.2 produces (by the product rule of differentiation) the following nonlinear form of the problem:

$$\begin{aligned}
 k \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \frac{dk}{du} \left(\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 \right) &= 0, & x \in (L_{int}, L_{ext}), y \in (0, L_{int}] \\
 & & x \in (L_{int}, L_{ext}), y \in (L_{int}, L_{ext}) \\
 & & x \in (0, L_{int}], y \in (L_{int}, L_{ext}) \\
 \frac{\partial u}{\partial y} &= 0, & x \in (L_{int}, L_{ext}), y = 0 \\
 u &= u_0, & x = L_{int}, y \in [0, L_{int}] \\
 u &= u_0, & x \in [0, L_{int}], y = L_{int} \\
 \frac{\partial u}{\partial x} &= 0, & x = 0, y \in (L_{int}, L_{ext}) \\
 -k \frac{\partial u}{\partial y} &= h(u - u_\infty), & x \in [0, L_{ext}], y = L_{ext} \\
 -k \frac{\partial u}{\partial x} &= h(u - u_\infty), & x = L_{ext}, y \in [0, L_{ext}],
 \end{aligned} \quad (4.3)$$

where $k = au^2 + bu + c$ and thus $\frac{dk}{du} = 2au + b$. This form of the problem is shown schematically in Fig. 4.3.

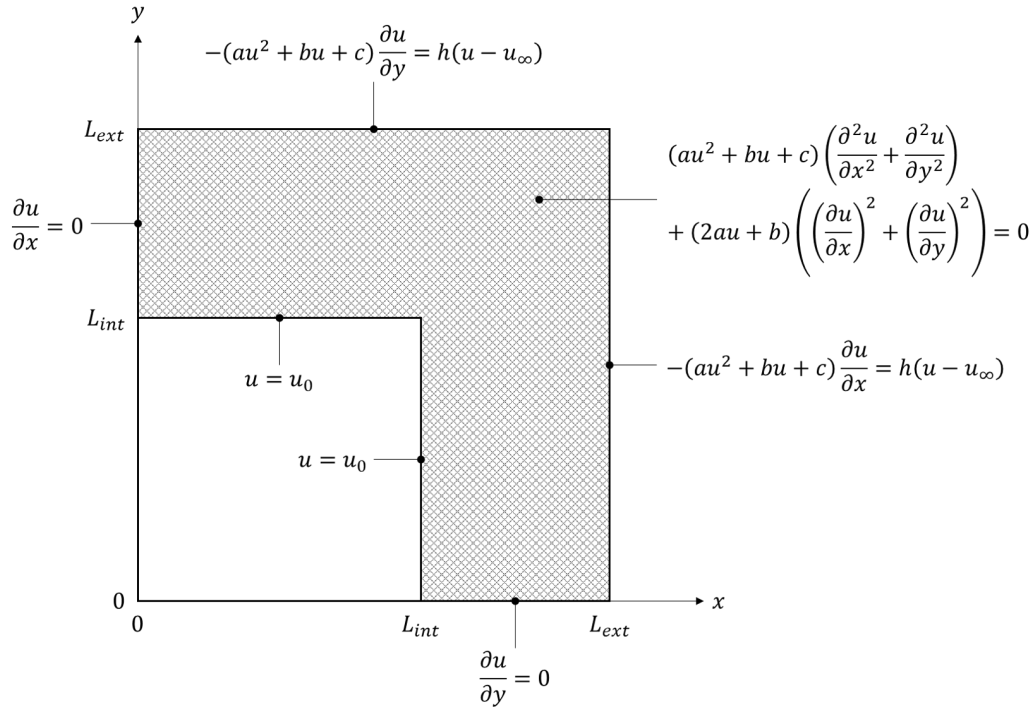


FIGURE 4.3: The nonlinear, dimensional form of the problem.

The values assigned to the dimensional parameters L_{int} , L_{ext} , u_0 , u_∞ , a , b , c , and h were chosen based upon realistic furnace dimensions, operating conditions, and insulation properties. These dimensional parameter values are shown in Table 4.1, where:

- The values assigned to L_{int} and L_{ext} are reasonable dimensions for this type of furnace [27], [33];
- The value assigned to u_0 is a reasonable setpoint temperature for this type of furnace [27], [33];
- The value assigned to u_∞ is room temperature, which is a conservative ambient temperature to assume from the perspective of furnace heat loss;
- The values assigned to a , b , and c are the quadratic equation coefficients computed by applying quadratic regression to manufacturer-provided thermal conductivity data for the eight commercially available high-temperature insulation materials listed in Table 4.2 [28]–[32]; and
- The value assigned to h is a reasonable estimate of the average convection HTC along the external surfaces of a furnace insulation module. Justification for selecting this value is provided in Appendix I using empirical correlations.

Parameter	Value(s)
L_{int} (m)	0.120
L_{ext} (m)	0.200
u_0 ($^{\circ}\text{K}$)	1273
u_{∞} ($^{\circ}\text{K}$)	293
a ($\frac{\text{W}}{\text{m}^{\circ}\text{K}^3}$)	$8.482 \times 10^{-8}, 8.482 \times 10^{-8}, 1.027 \times 10^{-7}, 1.206 \times 10^{-7},$ $9.780 \times 10^{-8}, 8.043 \times 10^{-8}, 4.974 \times 10^{-8}, 6.378 \times 10^{-8}$
b ($\frac{\text{W}}{\text{m}^{\circ}\text{K}^2}$)	$1.634 \times 10^{-5}, -3.366 \times 10^{-5}, -5.699 \times 10^{-5}, -4.936 \times 10^{-5},$ $-3.021 \times 10^{-5}, -2.052 \times 10^{-5}, 5.726 \times 10^{-5}, 3.047 \times 10^{-5}$
c ($\frac{\text{W}}{\text{m}^{\circ}\text{K}}$)	$3.221 \times 10^{-2}, 6.587 \times 10^{-2}, 1.429 \times 10^{-1}, 5.720 \times 10^{-2},$ $5.152 \times 10^{-2}, 8.180 \times 10^{-2}, 1.672 \times 10^{-1}, 1.993 \times 10^{-1}$
h ($\frac{\text{W}}{\text{m}^2\text{K}}$)	5

TABLE 4.1: The dimensional parameter space.

Material ID	Product Name	Manufacturer
1	Kaowool 1400LD	Morgan Advanced Materials
2	Kaowool 1400MD	Morgan Advanced Materials
3	Kaowool HS45	Morgan Advanced Materials
4	I-2600	Morgan Advanced Materials
5	I-2800	Morgan Advanced Materials
6	Alumina Type AL-30	ZIRCAR Ceramics
7	Alumina Type SALI	ZIRCAR Ceramics
8	Alumina Type SALI-2	ZIRCAR Ceramics

TABLE 4.2: Product name and manufacturer for each of the eight reference insulation materials [28]–[32].

The quadratic models used to approximate the thermal conductivities of the eight reference insulation materials, which were generated using Python’s NumPy library [34], are shown in Fig. 4.4-4.5. Collectively, these thermal conductivity models are assumed to be representative of the range of insulation materials typically found in this type of furnace.

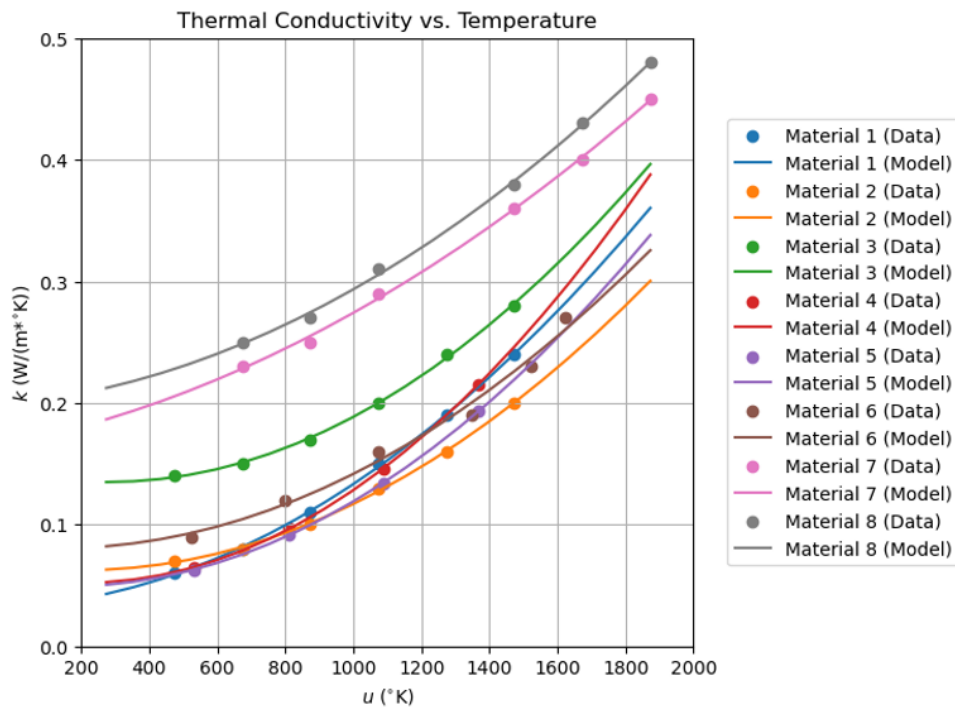


FIGURE 4.4: Tabulated temperature-dependent thermal conductivity data and corresponding best-fit quadratic equations for the eight reference insulation materials.

```

Material 1:
(a, b, c) = 8.482e-08, 1.634e-05, 3.221e-02
R^2 = 0.999
Material 2:
(a, b, c) = 8.482e-08, -3.366e-05, 6.587e-02
R^2 = 0.999
Material 3:
(a, b, c) = 1.027e-07, -5.699e-05, 1.429e-01
R^2 = 0.999
Material 4:
(a, b, c) = 1.206e-07, -4.936e-05, 5.720e-02
R^2 = 1.000
Material 5:
(a, b, c) = 9.780e-08, -3.021e-05, 5.152e-02
R^2 = 1.000
Material 6:
(a, b, c) = 8.043e-08, -2.052e-05, 8.180e-02
R^2 = 0.986
Material 7:
(a, b, c) = 4.974e-08, 5.726e-05, 1.672e-01
R^2 = 0.999
Material 8:
(a, b, c) = 6.378e-08, 3.047e-05, 1.993e-01
R^2 = 0.999

```

FIGURE 4.5: Computed quadratic equation coefficients and corresponding coefficients of determination for the eight reference insulation materials.

4.3 Dimensionless Form of the Problem

Thus far, the heat conduction problem has been considered in dimensional form. It is advantageous, however, to transform the problem into an equivalent dimensionless form prior to solving it, regardless of whether it is solved conventionally using numerical methods or using a neural network. Nondimensionalization is advantageous because it:

- Reduces the number of parameters in the problem and thus can reduce the effort required to solve the problem [35], [36];
- Reveals relationships between dimensional parameters that indicate how they collectively affect the problem's solution [35], [36];
- Enables the simultaneous solution of all dimensionally similar problem instances [35], [36]; and
- Can (but does not always) produce a normalized form of the problem, i.e., a form in which the dimensionless variables are all on the order of one [35].

The dimensional form of the problem shown in Eq. 4.3 and Fig. 4.3 involves $P = 11$ dimensional parameters: $u, x, y, L_{int}, L_{ext}, u_0, u_\infty, a, b, c,$ and h ; and $D = 3$ dimensions: length (m), temperature ($^\circ\text{K}$), and power (W). Per Buckingham's Pi Theorem, [35]–[37], the dimensional form of the problem can be reduced to an equivalent dimensionless form involving $P - D = 8$ dimensionless parameters. One equivalent form, obtained by applying the method of repeating variables [35] with $u_0, L_{ext},$ and c as repeating parameters, is:

$$\begin{aligned}
 \kappa \left(\frac{\partial^2 u^*}{\partial x^{*2}} + \frac{\partial^2 u^*}{\partial y^{*2}} \right) + \frac{d\kappa}{du^*} \left(\left(\frac{\partial u^*}{\partial x^*} \right)^2 + \left(\frac{\partial u^*}{\partial y^*} \right)^2 \right) &= 0, & x^* \in (\lambda, 1), y^* \in (0, \lambda] \\
 & & x^* \in (\lambda, 1), y^* \in (\lambda, 1) \\
 & & x^* \in (0, \lambda], y^* \in (\lambda, 1) \\
 \frac{\partial u^*}{\partial y^*} &= 0, & x^* \in (\lambda, 1), y^* = 0 \\
 u^* &= 1, & x^* = \lambda, y^* \in [0, \lambda] \\
 u^* &= 1, & x^* \in [0, \lambda], y^* = \lambda \\
 \frac{\partial u^*}{\partial x^*} &= 0, & x^* = 0, y^* \in (\lambda, 1) \\
 -\kappa \frac{\partial u^*}{\partial y^*} &= \eta(u^* - v), & x^* \in [0, 1], y^* = 1 \\
 -\kappa \frac{\partial u^*}{\partial x^*} &= \eta(u^* - v), & x^* = 1, y^* \in [0, 1],
 \end{aligned} \tag{4.4}$$

where $\kappa = \alpha u^{*2} + \beta u^* + 1$; $\frac{d\kappa}{du^*} = 2\alpha u^* + \beta$; and $x^* = \frac{x}{L_{ext}}, y^* = \frac{y}{L_{ext}}, \lambda = \frac{L_{int}}{L_{ext}}, v = \frac{u_\infty}{u_0}, \alpha = \frac{au_0^2}{c}, \beta = \frac{bu_0}{c}, \eta = \frac{hL_{ext}}{c}$, and $u^* = \frac{u}{u_0}$ are the eight dimensionless parameters remaining after nondimensionalization. This dimensionless form of the problem is shown schematically in Fig. 4.6.

The values of the dimensionless parameters $\lambda, v, \alpha, \beta,$ and η considered in this work are shown in Table 4.3. These correspond directly to the values of the dimensional parameters shown in Table 4.1. In combination with Eq. 4.4, these dimensionless parameter values fully define the parametric heat conduction problem considered in this work.

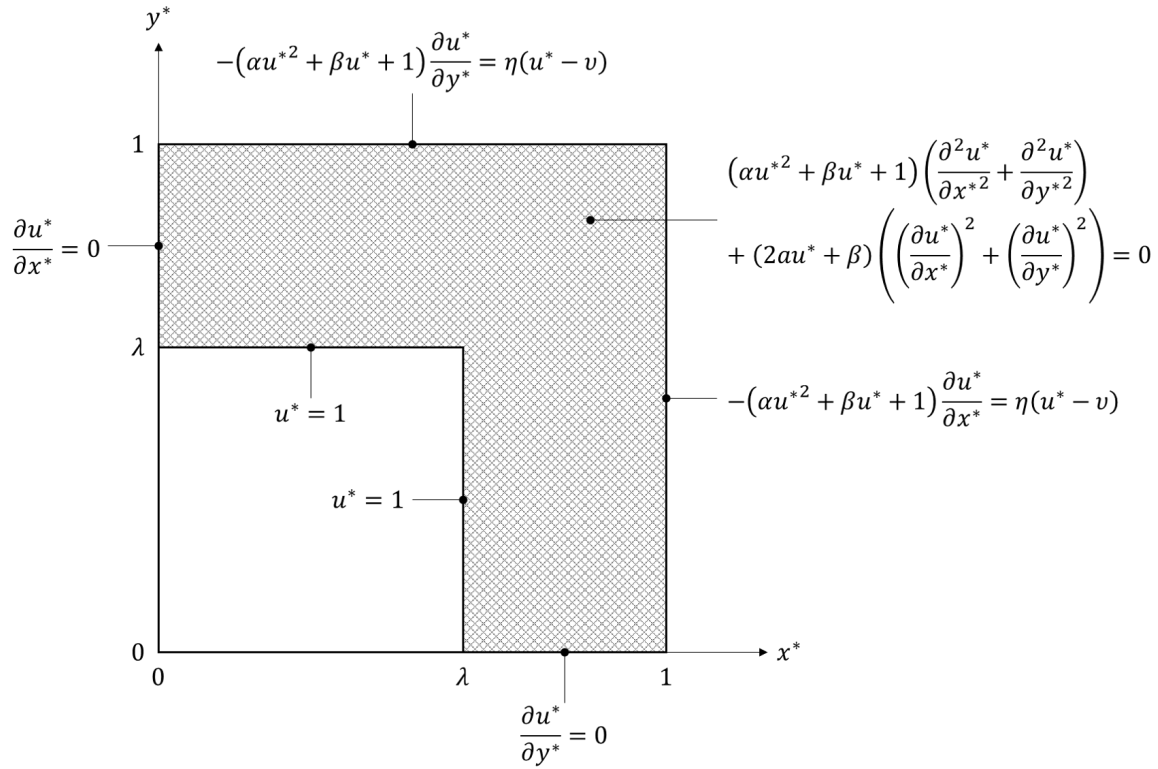


FIGURE 4.6: The nonlinear, dimensionless form of the problem.

Parameter	Value(s)
λ (1)	0.6
v (1)	0.230
α (1)	4.268, 2.087, 1.164, 3.416, 3.077, 1.593, 0.482, 0.519
β (1)	0.646, -0.651, -0.508, -1.098, -0.746, -0.319, 0.436, 0.195
η (1)	31.048, 15.182, 6.998, 17.482, 19.412, 12.225, 5.982, 5.018

TABLE 4.3: The dimensionless parameter space.

5 Methodology & Implementation

This chapter presents the methods employed to generate the testing dataset, which is used to evaluate the accuracy of the PINN, and describes the architecture, configuration, and training regimen of the PINN used to solve the problem.

5.1 Generation of Testing Data

The PINN presented in this work was trained without any training data; however, testing data was still required to evaluate the PINN's prediction accuracy. The testing dataset was generated by applying the FDM to solve each of eight problem instances described by the combination of Eq. 4.4 and the parameter values in Table 4.3. This was accomplished by:

1. Discretizing the spatial domain by creating a grid composed of $N \times N$ grid points $x_{i,j}^*, y_{i,j}^*$ with a uniform grid spacing of $\Delta x^* = \Delta y^*$, where $i \in \{0, \dots, N-1\}$, $j \in \{0, \dots, N-1\}$, and $\Delta x^* = \Delta y^* = \frac{1}{N-1}$. To simplify the implementation of the grid, a square grid spanning $x^* \in [0, 1]$, $y^* \in [0, 1]$ was used, which spans the domain of interest and the interior region it encloses (the region representative of the furnace chamber).
2. Formulating a set of finite difference equations, written in residual form, to approximate the differential equation and associated boundary conditions. The finite difference equation applicable to the domain nodes was derived by replacing the derivatives in the differential equation with centered, finite difference approximations with truncation error proportional to $\Delta x^{*2} = \Delta y^{*2}$. The finite difference equations applicable to the boundary nodes (both edge nodes and corner nodes) were derived by applying the law of conservation of energy to each physically unique node along the perimeter of the domain. This approach results in the replacement of derivatives with one-sided (i.e., forward/backward) finite difference approximations with truncation error proportional to $\Delta x^* = \Delta y^*$.¹
3. Using a nonlinear solver to compute the discrete solution $u_{i,j}^*$ at each grid point $x_{i,j}^*, y_{i,j}^*$.
4. Extracting and aggregating the discrete solutions that lie in the domain of interest to form the complete numerical solution.

This finite difference scheme was implemented using Python's NumPy library [34] and solutions were computed using a nonlinear solver in Python's SciPy library [38]. The complete set

¹The imposed temperature boundary conditions applied to the internal boundaries do not contain derivatives, so the finite difference equations applied to the internal boundary nodes are exact.

of finite difference equations used to compute these numerical solutions is shown in Appendix II, and the solutions obtained are shown in Fig. 5.1.

Because the solutions computed using this numerical model were used as testing data to evaluate the prediction accuracy of the PINN presented in the following chapter, significant effort was devoted to verifying their accuracy. First, a grid refinement test was conducted to confirm the grid independence of each FDM solution. Then, each FDM solution was compared to a corresponding solution computed using the FEM. These FEM solutions were computed using COMSOL Multiphysics, a well-known, commercially available multiphysics simulation software package.

5.2 Architecture of the PINN

The PINN used to solve this problem is a five-input, one-output FNN composed of 6 layers with 80 nodes per hidden layer. The hidden nodes apply the hyperbolic tangent activation function, and the output node applies the identity activation function, i.e., no activation function. The hidden layers therefore output:

$$\mathbf{a}^{(i)} = \frac{e^{\mathbf{z}^{(i)}} - e^{-\mathbf{z}^{(i)}}}{e^{\mathbf{z}^{(i)}} + e^{-\mathbf{z}^{(i)}}}, \quad (5.1)$$

where $\mathbf{z}^{(i)} = \mathbf{W}^{(i)}\mathbf{a}^{(i-1)} + \mathbf{b}^{(i)}$, and the output layer therefore outputs:

$$a^{(L)} = z^{(L)}, \quad (5.2)$$

where $z^{(L)} = \mathbf{w}^{(L)}\mathbf{a}^{(L-1)} + b^{(L)}$.

This neural network architecture was determined empirically with the competing objectives of having sufficient *expressivity* to output accurate predictions, while not being so large that training requires a prohibitively large amount of computing resources and/or time. Publications in the literature, particularly [8], [18], were helpful in identifying a reasonable initial architecture of approximately 5-10 layers and 20-200 hyperbolic tangent units per hidden layer.

To simplify the process of generating collocation points over the irregularly shaped spatial domain, it was partitioned into three rectangular subdomains, as shown in Fig. 5.2. These subdomains were defined geometrically as:

- Subdomain 1: $x^* \in (\lambda, 1), y^* \in (0, \lambda]$;
- Subdomain 2: $x^* \in (\lambda, 1), y^* \in (\lambda, 1)$; and
- Subdomain 3: $x^* \in (0, \lambda], y^* \in (\lambda, 1)$.

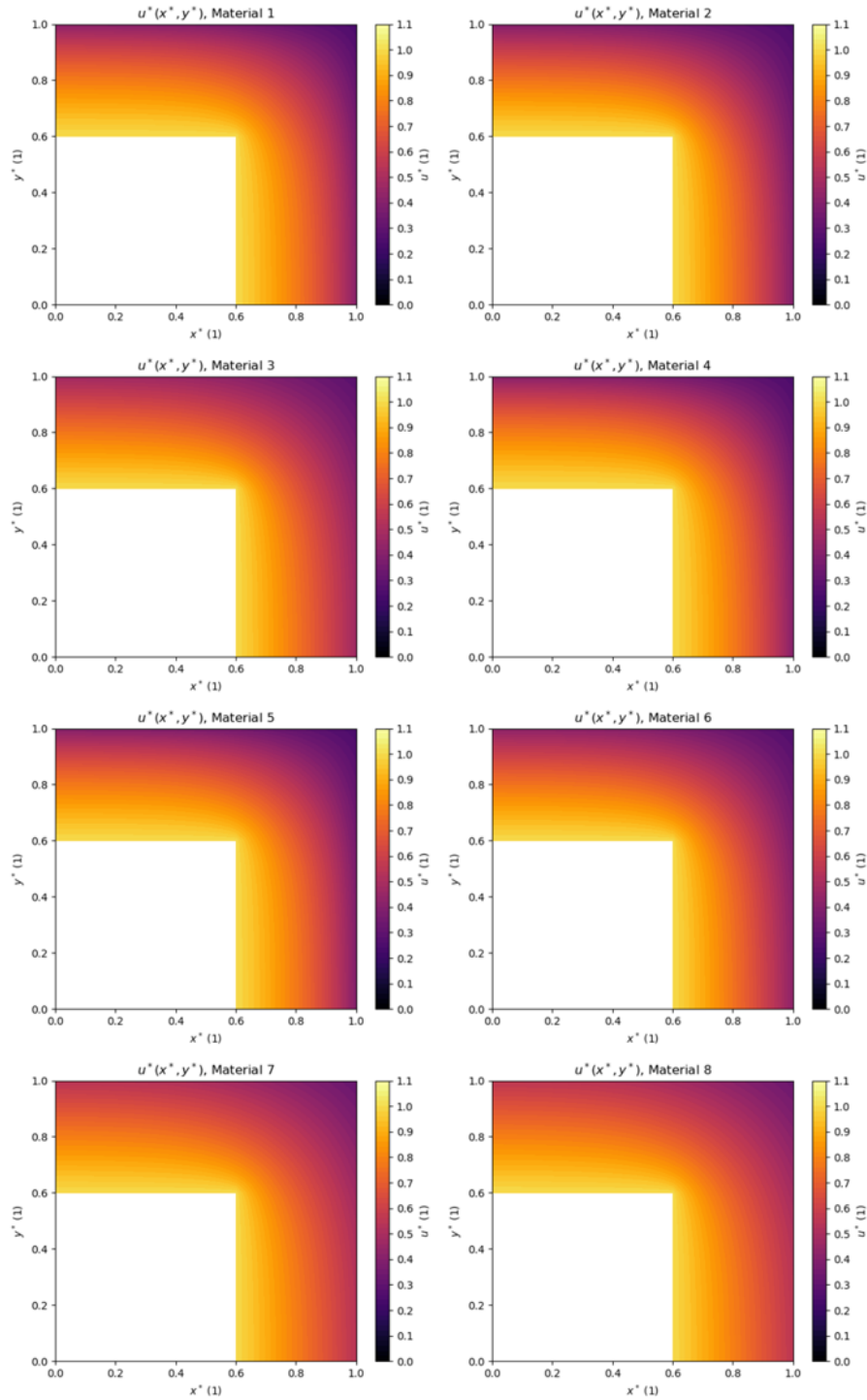


FIGURE 5.1: Numerical solution $u^*(x^*, y^*)$ for each problem instance, i.e., each reference material, included in the testing dataset.

The MSE loss function was used to compute the physics-based loss components; however, instead of computing the composite loss as the sum of aggregated domain and boundary losses as shown in Eq. 3.5, it was computed as the sum of individual subdomain and boundary losses. Configuring the composite loss in this way does add some complexity from a formulation perspective; however, weighting each subdomain loss \mathcal{L}_{D_i} and boundary loss \mathcal{L}_{B_j} enables the loss on each subdomain and boundary to be penalized individually, which may provide better training results in some instances.²

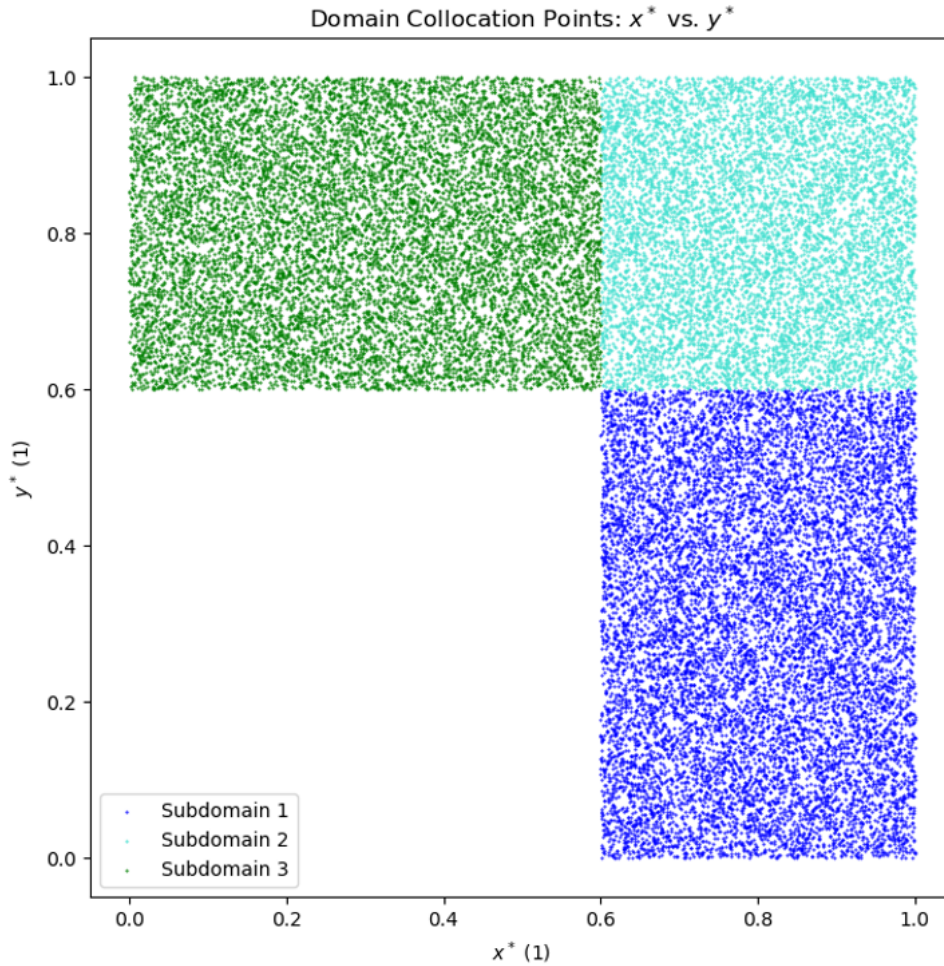


FIGURE 5.2: Domain collocation points projected onto 2-D plane defined by x^* , y^* .

The composite loss was therefore computed as:

$$\mathcal{L} = \sum_{i=1}^{N_D} (\lambda_{D_i} \mathcal{L}_{D_i}) + \sum_{j=1}^{N_B} (\lambda_{B_j} \mathcal{L}_{B_j}), \quad (5.3)$$

²This approach was used in this work, but it was not rigorously studied.

where $\lambda_{D_i}\mathcal{L}_{D_i}$ is the weighted loss on subdomain i , N_D is the number of subdomains, $\lambda_{B_j}\mathcal{L}_{B_j}$ is the weighted loss on boundary j , and N_B is the number of boundaries. The subdomain loss \mathcal{L}_{D_i} and boundary loss \mathcal{L}_{B_j} in Eq. 5.3 were computed as:

$$\begin{aligned}\mathcal{L}_{D_i} &= \frac{1}{N_{D_i}} \sum_{i=1}^{N_{D_i}} \left(\mathcal{R}_{D_i}^{[k]^2} \right) \\ \mathcal{L}_{B_j} &= \frac{1}{N_{B_j}} \sum_{i=1}^{N_{B_j}} \left(\mathcal{R}_{B_j}^{[k]^2} \right),\end{aligned}\tag{5.4}$$

respectively, where $\mathcal{R}_{D_i}^{[k]}$ is the residual associated with a given collocation point k on subdomain i , N_{D_i} is the number collocation points on subdomain i , $\mathcal{R}_{B_j}^{[k]}$ is the residual associated with a given collocation point k on boundary j , and N_{B_j} is the number of collocation points on boundary j .

The differential equation and boundary condition residuals $\mathcal{R}_{D_i}^{[k]}$ and $\mathcal{R}_{B_j}^{[k]}$ in Eq. 5.4 were therefore computed as:³

$$\begin{aligned}\mathcal{R}_{D1} &= \kappa \left(\frac{\partial^2 \hat{u}^*}{\partial x^{*2}} + \frac{\partial^2 \hat{u}^*}{\partial y^{*2}} \right) + \frac{d\kappa}{d\hat{u}^*} \left(\left(\frac{\partial \hat{u}^*}{\partial x^*} \right)^2 + \left(\frac{\partial \hat{u}^*}{\partial y^*} \right)^2 \right), & x^* \in (\lambda, 1), y^* \in (0, \lambda] \\ \mathcal{R}_{D2} &= \kappa \left(\frac{\partial^2 \hat{u}^*}{\partial x^{*2}} + \frac{\partial^2 \hat{u}^*}{\partial y^{*2}} \right) + \frac{d\kappa}{d\hat{u}^*} \left(\left(\frac{\partial \hat{u}^*}{\partial x^*} \right)^2 + \left(\frac{\partial \hat{u}^*}{\partial y^*} \right)^2 \right), & x^* \in (\lambda, 1), y^* \in (\lambda, 1) \\ \mathcal{R}_{D3} &= \kappa \left(\frac{\partial^2 \hat{u}^*}{\partial x^{*2}} + \frac{\partial^2 \hat{u}^*}{\partial y^{*2}} \right) + \frac{d\kappa}{d\hat{u}^*} \left(\left(\frac{\partial \hat{u}^*}{\partial x^*} \right)^2 + \left(\frac{\partial \hat{u}^*}{\partial y^*} \right)^2 \right), & x^* \in (0, \lambda], y^* \in (\lambda, 1) \\ \mathcal{R}_{B1} &= \frac{\partial \hat{u}^*}{\partial y^*}, & x^* \in (\lambda, 1), y^* = 0 \\ \mathcal{R}_{B2} &= \hat{u}^* - 1, & x^* = \lambda, y^* \in [0, \lambda] \\ \mathcal{R}_{B3} &= \hat{u}^* - 1, & x^* \in [0, \lambda], y^* = \lambda \\ \mathcal{R}_{B4} &= \frac{\partial \hat{u}^*}{\partial x^*}, & x^* = 0, y^* \in (\lambda, 1) \\ \mathcal{R}_{B5} &= \kappa \frac{\partial \hat{u}^*}{\partial y^*} + \eta(\hat{u}^* - v), & x^* \in [0, 1], y^* = 1 \\ \mathcal{R}_{B6} &= \kappa \frac{\partial \hat{u}^*}{\partial x^*} + \eta(\hat{u}^* - v), & x^* = 1, y^* \in [0, 1],\end{aligned}\tag{5.5}$$

where $\kappa = \alpha \hat{u}^{*2} + \beta \hat{u}^* + 1$ and $\frac{d\kappa}{d\hat{u}^*} = 2\alpha \hat{u}^* + \beta$.

Combining Eq. 5.3-5.5 with the previously described FNN structure yields the PINN used to solve the problem. This PINN, which was implemented, trained, and tested using Python's PyTorch library [6], is shown schematically in Fig. 5.3.

5.3 Training the PINN

The PINN was trained using a total of 36800 collocation points, each of the form $x^{*[k]}, y^{*[k]}, \alpha^{[k]}, \beta^{[k]}, \eta^{[k]}$. More specifically, the PINN was trained with $N_{D_1} = 12000$, $N_{D_2} = 8000$, $N_{D_3} = 12000$, and $N_{B_1}, \dots, N_{B_6} = 800$. Each set of collocation points was generated by uniform sampling (i.e., randomly sampling a uniform distribution over) the applicable 5-D parameter

³The superscript $[k]$, which indicates an association with a collocation point k , is implied in Eq. 5.5 (instead of included explicitly as done in Eq. 3.7) to keep the length of the equations reasonable.

space, where the range of values for each independent variable x^* and y^* was determined by the possible range of x^* and y^* values in or along the subdomain or boundary respectively, and the range of values for each parameter α , β , and η was taken directly from Table 4.3. The resultant sets of collocation points can be partially visualized in the 3-D plots shown in Fig. 5.4-5.5.⁴

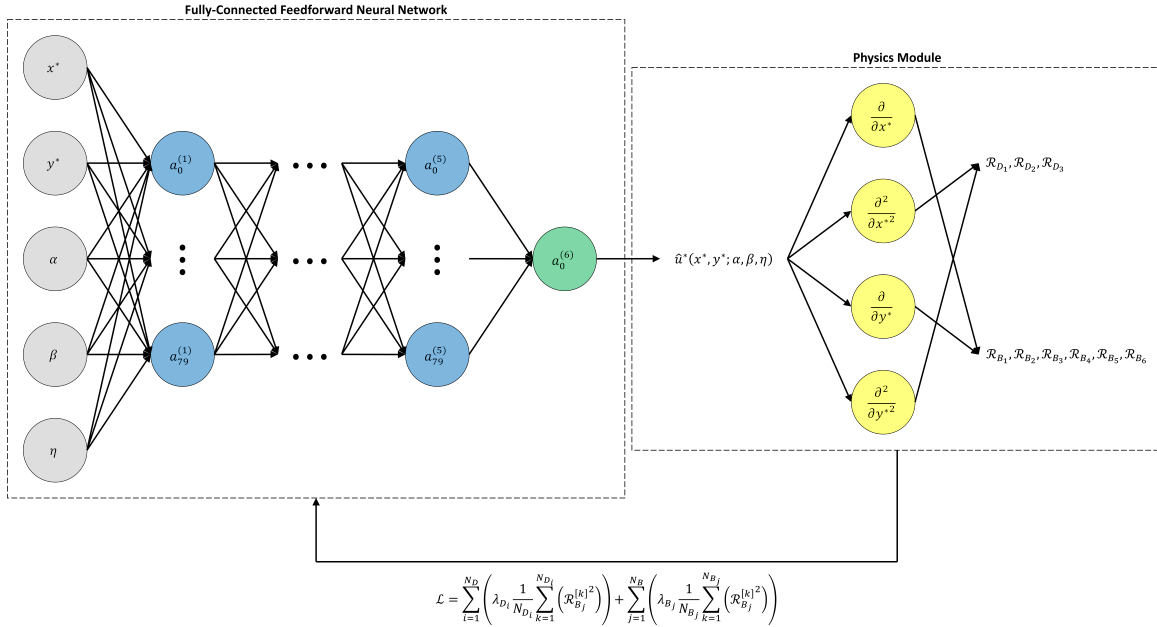


FIGURE 5.3: Schematic of the PINN used to approximate the solution to the problem of interest.

A subtle but noteworthy implication of generating collocation points in this manner is that each collocation point contains a combination of randomly sampled α , β , and η values. Consequently, while the range of each parameter α , β , and η in the testing dataset was limited to corresponding range of values in Table 4.3, the effective range of thermal conductivity vs. temperature curves used in the training process was comparatively broad, as shown in Fig. 5.6.

⁴Fully visualizing these sets of collocation points would require 5-D plots.

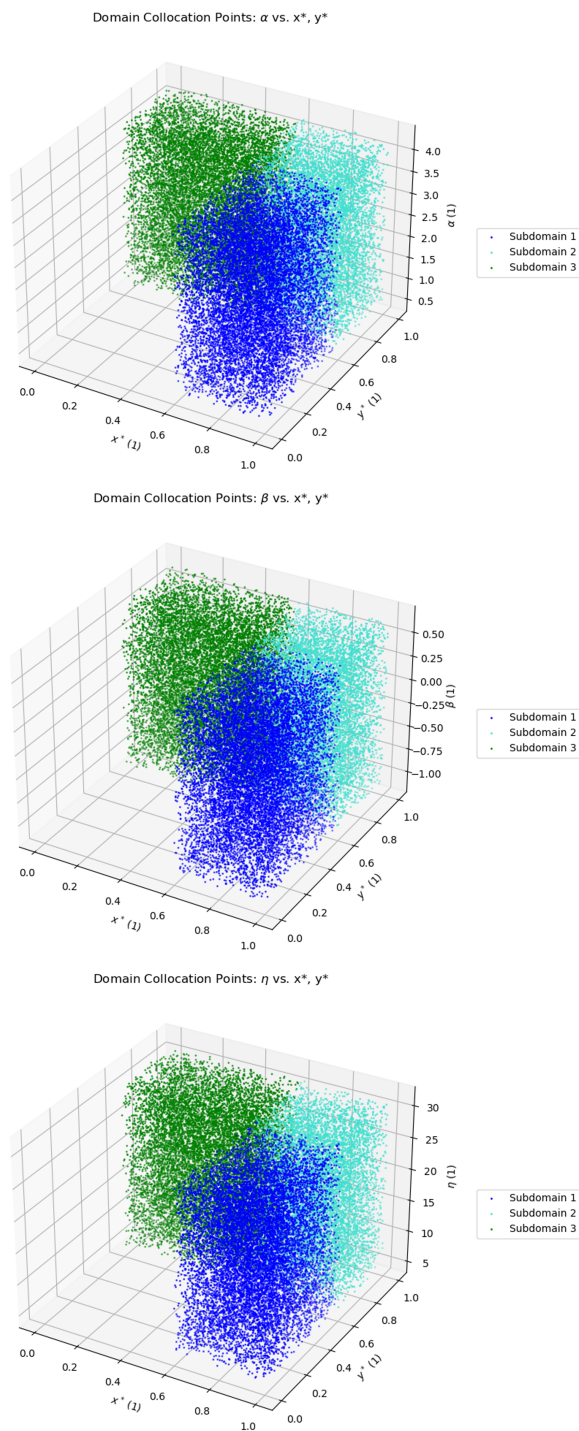


FIGURE 5.4: Domain collocation points projected onto the 3-D spaces defined by x^* , y^* , α (top), x^* , y^* , β (middle), and x^* , y^* , η (bottom).

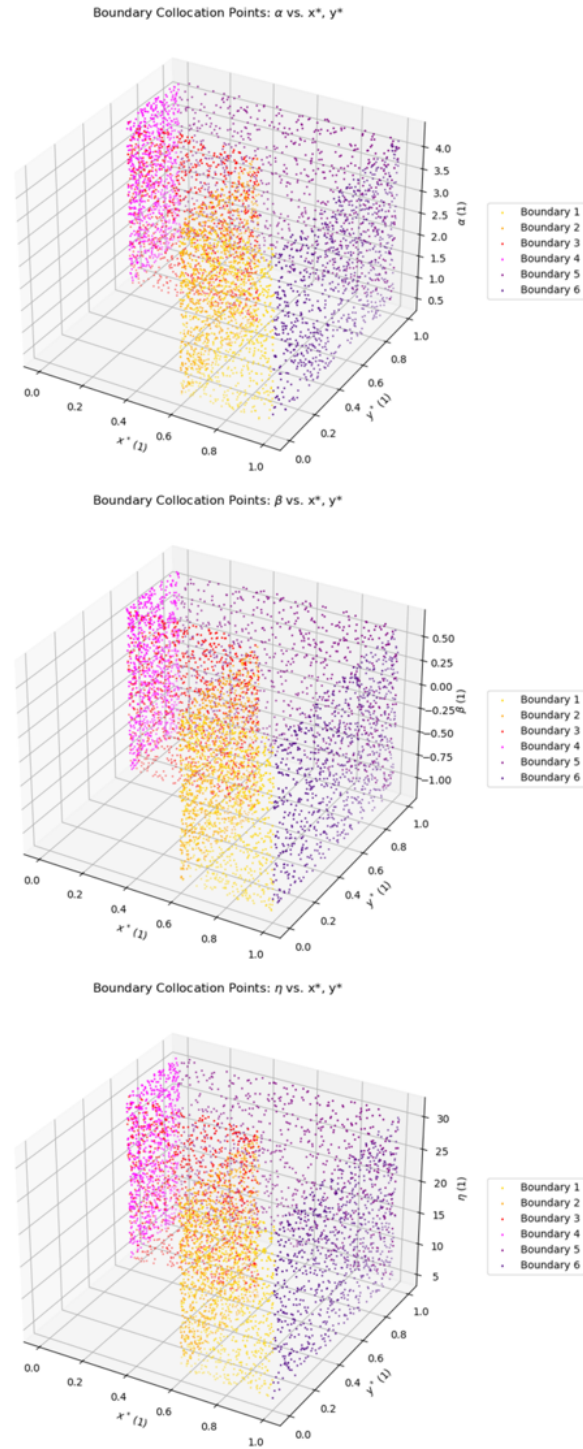


FIGURE 5.5: Boundary collocation points projected onto the 3-D spaces defined by x^* , y^* , α (top), x^* , y^* , β (middle), and x^* , y^* , η (bottom).

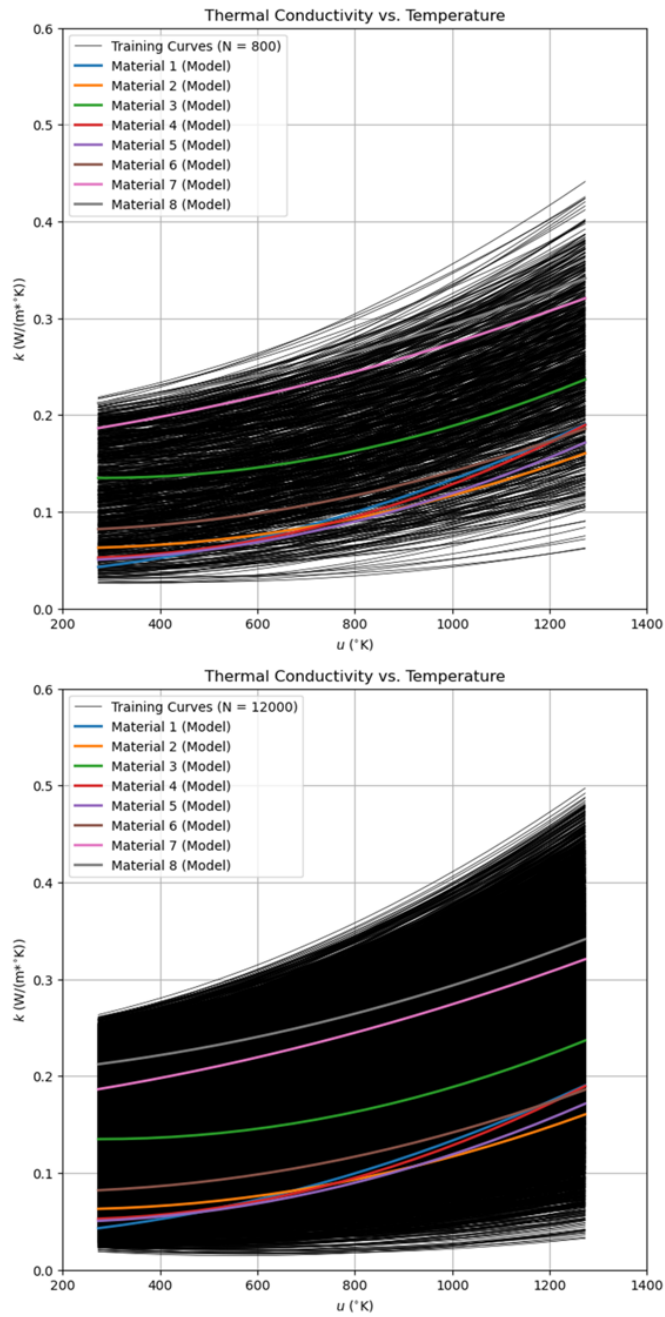


FIGURE 5.6: Thermal conductivity vs. temperature curves associated with collocation points generated by uniformly sampling the dimensionless parameter space specified in Table 4.3. The number of training curves (black) included in the top plot is 800, the minimum number of collocation points assigned to any subdomain/boundary, and the number included in the bottom plot is 12000, the maximum number of collocation points assigned to any subdomain/boundary.

The weight values used to compute the composite loss in Eq. 5.3 were determined empirically. This approach was chosen over alternative approaches, such as estimating optimal weights before training [39] or adapting weights during training [40], due to its relative simplicity. The final weights used are shown in Table 5.1.

Weight	Value
λ_{D1}	0.3
λ_{D2}	0.3
λ_{D3}	0.3
λ_{B1}	0.1
λ_{B2}	1.0
λ_{B3}	1.0
λ_{B4}	0.1
λ_{B5}	0.1
λ_{B6}	0.1

TABLE 5.1: The weights used to compute the PINN’s composite loss.

The neural network parameters θ were optimized using the Adam optimizer, an extension of the standard stochastic gradient descent algorithm that applies unique, adaptive learning rates to each parameter based upon recent values of the gradient computed for that parameter [41]. This optimizer was chosen over alternatives because of its combined effectiveness and ease of use. A three-step training process was used consisting of:

1. 50,000 iterations with an initial learning rate of 1×10^{-3} (default value),
2. 50,000 iterations with an initial learning rate of 1×10^{-4} , and
3. 50,000 iterations with an initial learning rate of 1×10^{-5} .

This three-step training process is similar to the multi-step training process used in [18].

6 Results

This chapter compares the PINN's solution predictions to the true solutions contained in the testing dataset. The error in each solution prediction, as well as the aggregate error over the entire testing dataset, is quantified and discussed.

6.1 Presentation of Results

After training, the PINN's accuracy was evaluated using the testing dataset presented in the previous chapter. The plots in Fig. 6.1 show the PINN's solution prediction for each of the problem instances in the testing dataset, i.e., for each of the eight reference insulation materials listed in Table 4.1. Corresponding plots showing the error in each solution prediction (relative to the corresponding true solution shown in Fig. 5.1) are shown in Fig. 6.2.

To quantify the accuracy of the PINN more generally, the relative root mean square error (RRMSE) was computed for each problem instance in the testing dataset, as well as for the full testing dataset. The RRMSE, which normalizes the RMS error to the RMS true value, is an effective metric for evaluating and comparing the performance of predictive models and is computed as [42], [43]:

$$\mathcal{L}_{RRMSE} = \sqrt{\frac{1}{N} \sum_{k=1}^N \left(\frac{(u^{*[k]} - \hat{u}^{*[k]})^2}{u^{*[k]2}} \right)}, \quad (6.1)$$

where $u^{*[k]}$ and $\hat{u}^{*[k]}$ are the true and predicted values associated with a given point k in the testing dataset; and N is the number of values considered from the dataset. The resultant RRMSE values are shown in Table 6.1.

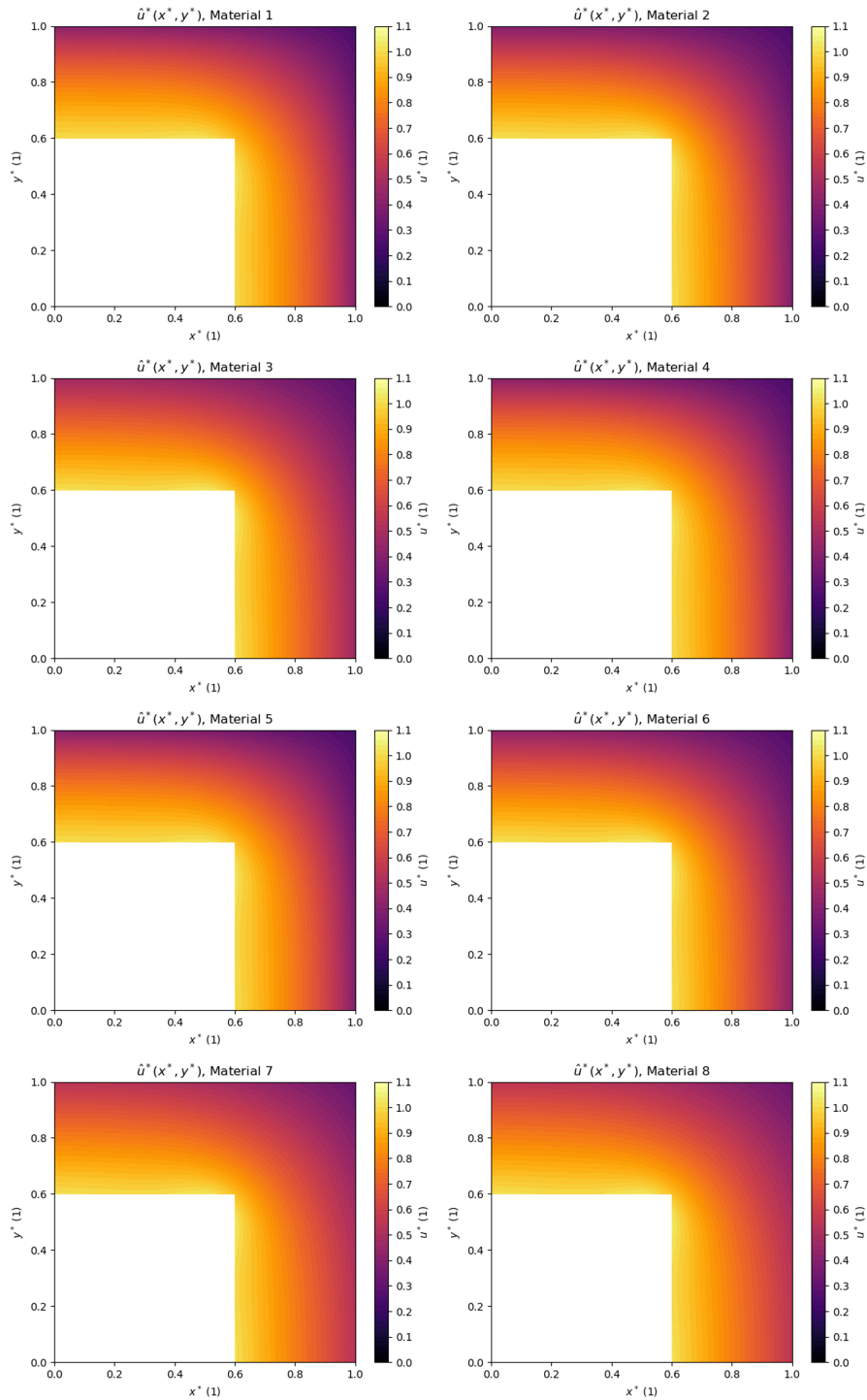


FIGURE 6.1: Solution prediction $\hat{u}^*(x^*, y^*)$ for each problem instance, i.e., each reference material, included in the testing dataset.

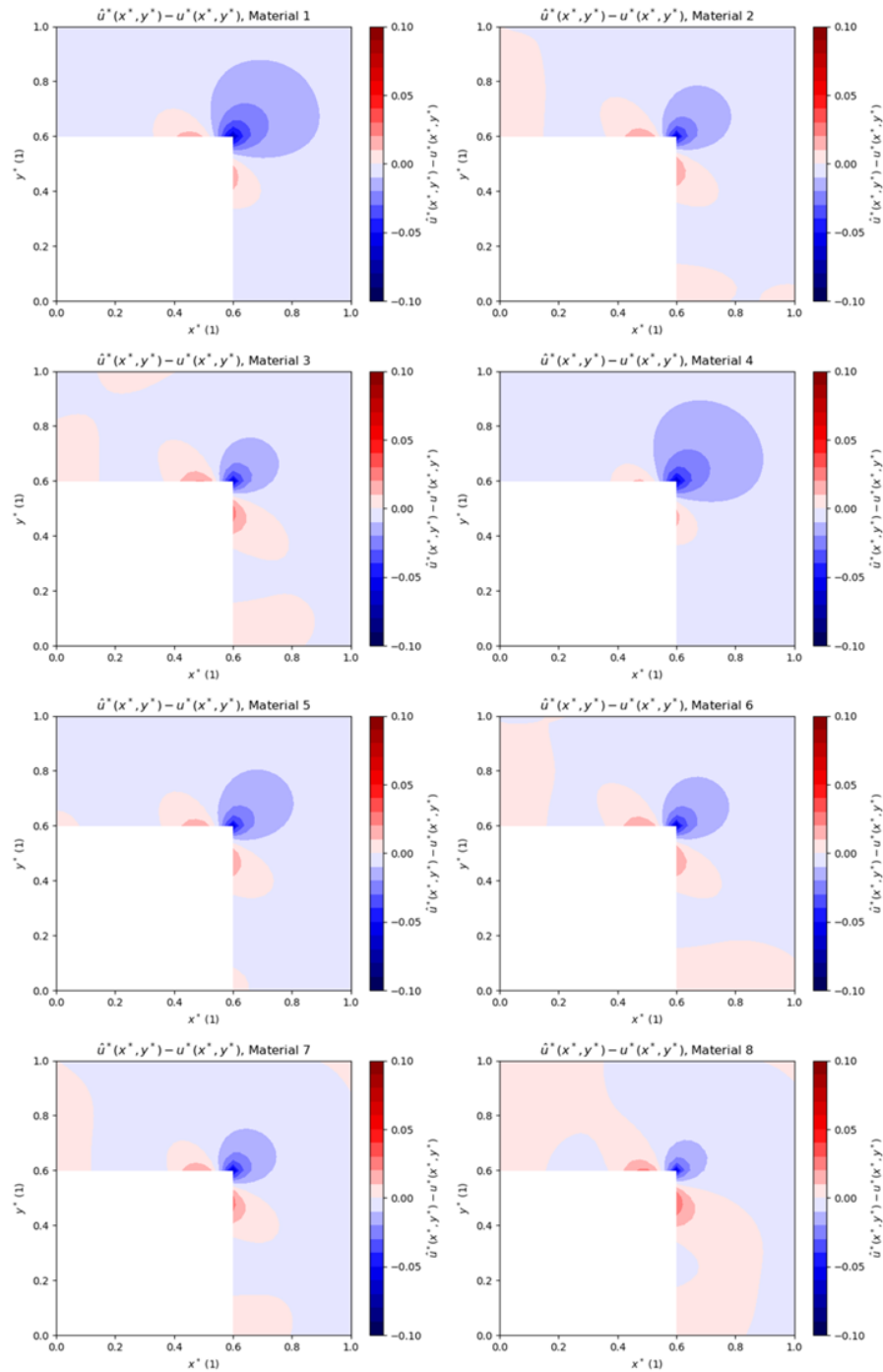


FIGURE 6.2: Error $\hat{u}^*(x^*, y^*) - u^*(x^*, y^*)$ for each problem instance, i.e., each reference material, included in the testing dataset.

Problem Instance (Material ID)	RRMSE
1	1.138×10^{-2}
2	7.699×10^{-3}
3	7.148×10^{-3}
4	1.150×10^{-2}
5	8.199×10^{-3}
6	7.684×10^{-3}
7	6.367×10^{-3}
8	5.334×10^{-3}
All	9.063×10^{-3}

TABLE 6.1: RRMSE of each solution prediction.

6.2 Discussion of Results

As shown in Table 6.1, the PINN's solution predictions were generally accurate across the entire testing dataset. The aggregate RRMSE was approximately 0.91% and the worst-case individual RRMSE (associated with Material 4) was approximately 1.15%. Six of the eight individual RRMSE values were in the range 0.53-0.82%.

In each problem instance, the highest magnitude error occurred along the internal boundaries at which the imposed temperature boundary condition is applied, particularly at/near the intersection of these two boundaries. This is illustrated in Fig. 6.3-6.4 using the solution prediction associated with Material 1 as an example (corresponding plots for Materials 2-8 are provided in Appendix III). From a physics perspective, this observed tendency is not surprising because the magnitude of the temperature gradient is highest in this area.

Similarly, the RRMSE was generally higher in the problem instances associated with materials that have lower and more variable thermal conductivities over the relevant temperature range. Again, this is not surprising from a physics perspective because:

- The temperature gradient in a material with a low thermal conductivity will generally be larger than the temperature gradient in a material with a higher thermal conductivity.
- The maximum value of the temperature gradient will generally be higher in a material with a highly temperature-dependent thermal conductivity averaging \bar{k} over the temperature range of interest than a material with a constant thermal conductivity equal to \bar{k} .

Regarding the time and computing resources required to obtain these results, the duration of the training process was approximately 3 hours using a Dell Precision 5680 mobile workstation with an Intel i9-139000H CPU and NVIDIA RTX 2000 Ada Laptop GPU. Given the general availability of desktop workstations with multiple, significantly more powerful GPUs, this training time could presumably be reduced significantly while still using a single, off-the-shelf machine.

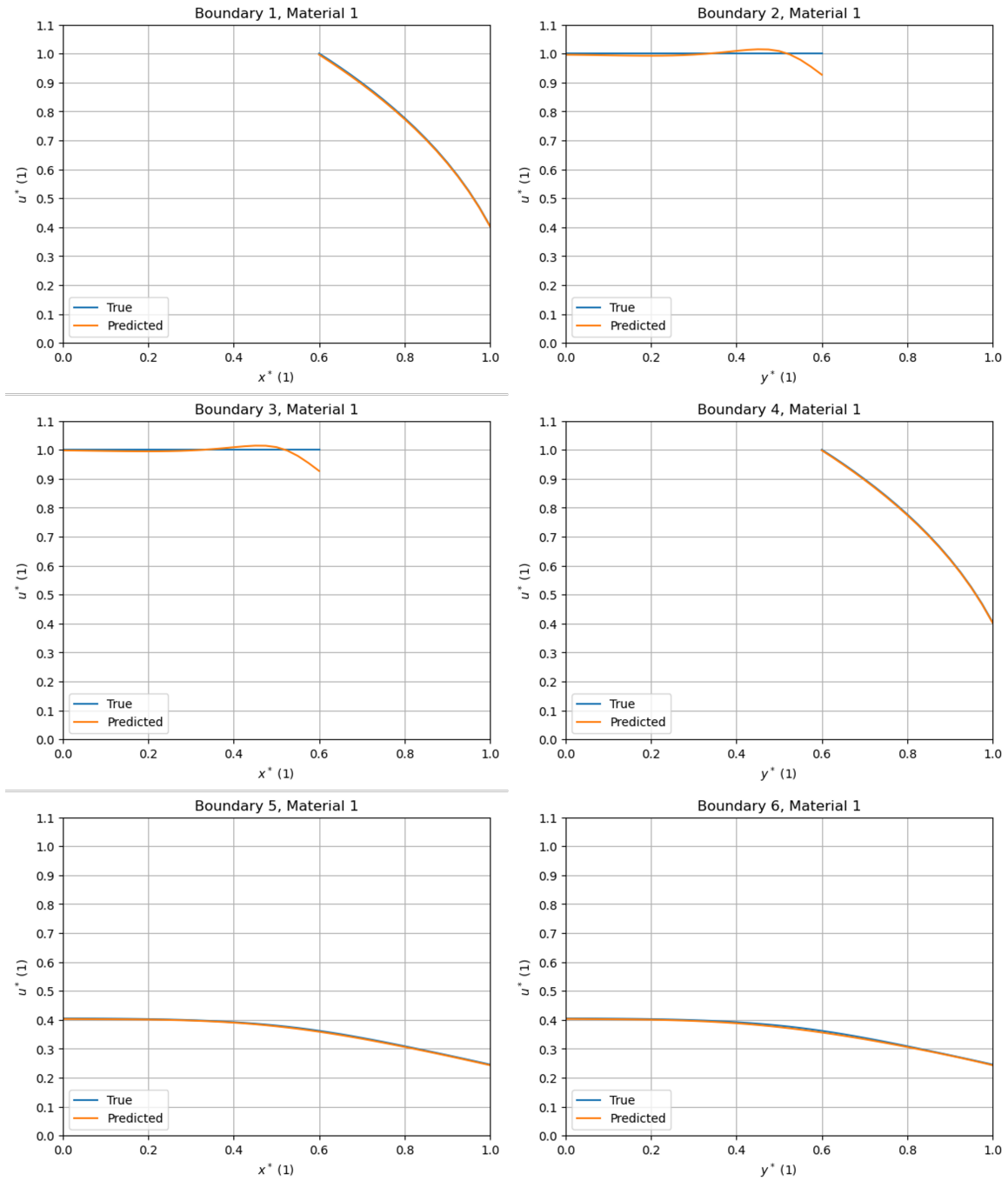


FIGURE 6.3: Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 1. A key for the boundary numbering scheme used is provided in Fig. 6.4.

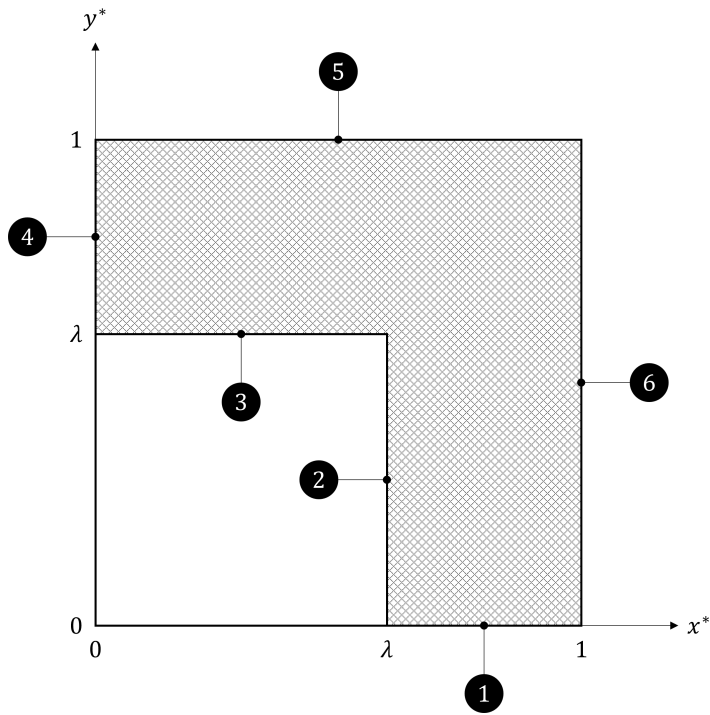


FIGURE 6.4: Key for the boundary numbering scheme used in Fig. 6.3.

7 Conclusion

This chapter summarizes the contributions that this work makes to the computational physics and physics-focused machine learning fields and identifies opportunities for future research.

7.1 Contributions of this Work

The results presented in the previous chapter demonstrate the feasibility of training a PINN, without using any training data, to accurately solve heat conduction problems involving parameterized, temperature-dependent material properties models. To the author's knowledge, this is the first time a PINN has been applied to the solution of any computational physics problem involving parameterized, nonlinear material property models, irrespective of whether data is used in the training process.

More generally, this work also shows how the Python programming language and existing open-source libraries can be applied to practically all aspects of implementing a PINN to solve this type of problem, including:

- The statistical analysis of empirically derived material property data and the creation of representative material property models;
- The generation of simulation-based testing data;
- The initialization and configuration of a PINN;
- The training of a PINN;
- The evaluation of a PINN's accuracy; and
- The postprocessing of a PINN's solution predictions.

7.2 Opportunities for Future Research

This work is far from exhaustive and presents many opportunities for future research, some of which are already active research areas. These opportunities for further research include, but are not limited to:

- Investigating how the results obtained in this work would be affected by (i) varying the architecture of the PINN, (ii) enforcing boundary conditions as "hard" constraints on the

solution [17], [25], [26]; (iii) generating collocation points using an alternative sampling method (e.g., Latin hypercube sampling [44]), (iv) adapting loss weights during the training process [39], and/or (v) using alternative optimization algorithms to update neural network parameters.

- Using PINNs to solve similar problems (i.e., problems that also involve parameterized, nonlinear material property models) that involve additional geometric and/or physical complexities. This includes 3-D problems, transient problems, multiphysics problems involving systems of coupled differential equations, etc.
- Using PINNs to solve similar problems in which the range of values considered for each parameter is significantly larger.
- Using PINNs to solve problems that involve multiple parameterized, nonlinear material properties, e.g., applying PINNs to solve a transient heat conduction problem in which temperature-dependent models are used for both the thermal conductivity and specific heat capacity of the heat-conducting medium.
- Using PINNs to solve problems that are also parameterized in other ways, e.g., problems that also involve parameterized geometries and/or boundary conditions.
- Investigating the practical limitations of using PINNs to solve parametric problems (e.g., the extent to which the *curse of dimensionality* limits the application of PINNs to parametric problems, and how this may be mitigated).
- Investigating if/how PINNs, or select components of the PINN framework (e.g., automatic differentiation), could be leveraged to accelerate the solution of computational physics problems using traditional numerical solution methods.

8 Appendix I

As specified in Chapter 4, the average convection HTC along the external surfaces of the furnace insulation module was assumed to be $5 \text{ W}/(\text{m}^2\text{°K})$. This value was derived from the calculations shown in Fig. 8.1-8.2, which estimate the natural convection HTC along each physically unique external surface of the furnace insulation module considered in the 2-D cross-section (i.e., the vertical side surfaces, the horizontal top surface, and the horizontal bottom surface). These calculations apply well-known empirical correlations of the form [1], [2]:

$$\bar{h}_L = \overline{Nu}_L \frac{k}{L_c} \quad (8.1)$$

where \bar{h}_L is the average HTC along the surface; \overline{Nu}_L is the average Nusselt number along the surface and is computed using empirically derived relationships based upon the configuration of the heat transfer problem along the surface; k is the thermal conductivity of air evaluated at the film temperature along the surface, and L_c is the characteristic length of the surface. The empirical correlations used to compute \overline{Nu}_L were taken from [2].

In these calculations:

- The length L of each surface was assumed to be 0.4 m, twice the half-length of L_{ext} ;
- The depth d of each surface was assumed to be 0.6 m, a reasonable assumption given the typical aspect ratios of commercially available furnaces [27], [33]; and
- All properties of air were computed assuming an ambient pressure of 1 atm.

Two different cases were considered; in the first case (Fig. 8.1), the average surface temperature u_s was assumed to be 440 °K (equivalent to a dimensionless temperature of $u_s^* = 0.346$), and in the second case (Fig. 8.2), u_s was assumed to be 610 °K (equivalent to a dimensionless temperature of $u_s^* = 0.479$). These values were selected deliberately to span the approximate range of average surface temperatures computed for the eight problem instances that comprise the testing data. As shown in Fig. 8.1-8.2, the calculated natural convection HTCs range from 3.82 to 8.76, confirming that the assumed value of $5 \text{ W}/(\text{m}^2\text{°K})$ is a reasonable estimate.

INPUT VALUE(S)	INTERMEDIATE VALUE(S)	OUTPUT VALUE(S)
u_s (°K) 440 < Average surface temperature u_∞ (°K) 293 < Ambient air temperature g (m/s ²) 9.81 < Gravitational acceleration L [m] 0.40 < Length of surface d [m] 0.60 < Depth of surface Orientation V < Orientation of surface	L_c (m) 0.40 u_f (°K) 366.5 < Film temperature $\beta(u_f)$ (1/°K) 2.73E-03 < Coeff. of volume expansion @ u_f $\nu(u_f)$ (m ² /s) 2.20E-05 < Kinematic viscosity of air @ u_f Gr_f (1) 5.20E+08 < Grashof number $Pr(u_f)$ (1) 0.713 < Prandtl number of air @ u_f Ra_f (1) 3.71E+08 < Rayleigh number $Nu_{L,avg}$ (1) 90.7 < Nusselt number (averaged over L) $k(u_f)$ (W/(m°K)) 3.02E-02 < Thermal conductivity of air @ u_f	$h_{L,avg}$ (W/(m ² °K)) 6.85 < HTC (averaged over L)
INPUT VALUE(S) u_s (°K) 440 < Average surface temperature u_∞ (°K) 293 < Ambient air temperature g (m/s ²) 9.81 < Gravitational acceleration L [m] 0.40 < Length of surface d [m] 0.60 < Depth of surface Orientation H, Upper < Orientation of surface	INTERMEDIATE VALUE(S) L_c (m) 0.12 u_f (°K) 366.5 < Film temperature $\beta(u_f)$ (1/°K) 2.73E-03 < Coeff. of volume expansion @ u_f $\nu(u_f)$ (m ² /s) 2.20E-05 < Kinematic viscosity of air @ u_f Gr_f (1) 1.40E+07 < Grashof number $Pr(u_f)$ (1) 0.713 < Prandtl number of air @ u_f Ra_f (1) 1.00E+07 < Rayleigh number $Nu_{L,avg}$ (1) 32.3 < Nusselt number (averaged over L) $k(u_f)$ (W/(m°K)) 3.02E-02 < Thermal conductivity of air @ u_f	OUTPUT VALUE(S) $h_{L,avg}$ (W/(m ² °K)) 8.13 < HTC (averaged over L)
INPUT VALUE(S) u_s (°K) 440 < Average surface temperature u_∞ (°K) 293 < Ambient air temperature g (m/s ²) 9.81 < Gravitational acceleration L [m] 0.40 < Length of surface d [m] 0.60 < Depth of surface Orientation H, Lower < Orientation of surface	INTERMEDIATE VALUE(S) L_c (m) 0.12 u_f (°K) 366.5 < Film temperature $\beta(u_f)$ (1/°K) 2.73E-03 < Coeff. of volume expansion @ u_f $\nu(u_f)$ (m ² /s) 2.20E-05 < Kinematic viscosity of air @ u_f Gr_f (1) 1.40E+07 < Grashof number $Pr(u_f)$ (1) 0.713 < Prandtl number of air @ u_f Ra_f (1) 1.00E+07 < Rayleigh number $Nu_{L,avg}$ (1) 15.2 < Nusselt number (averaged over L) $k(u_f)$ (W/(m°K)) 3.02E-02 < Thermal conductivity of air @ u_f	OUTPUT VALUE(S) $h_{L,avg}$ (W/(m ² °K)) 3.82 < HTC (averaged over L)

EQUATIONS APPLIED

Calculation of u_f :

$$u_f = \frac{u_s + u_\infty}{2}$$

Calculation of Ra_f :

$$Ra_f = Gr_f Pr = \frac{g \beta (u_s - u_\infty) L_c^3}{\nu^2} Pr$$

Calculation of $Nu_{L,avg}$ for vertical surfaces (Orientation = V)

$$\bar{Nu}_L = \left(0.825 + \frac{0.387 Ra_f^{1/4}}{\left(1 + \left(\frac{0.492}{Pr} \right)^{1/4} \right)^{1/4}} \right)^2$$

Calculation of $Nu_{L,avg}$ for horizontal, upper surface (Orientation = H, Upper):

$$\begin{cases} \bar{Nu}_L = 0.54 Ra_f^{1/4}, & 10^4 \leq Ra_f < 10^7 \\ \bar{Nu}_L = 0.15 Ra_f^{1/3}, & 10^7 \leq Ra_f \leq 10^{11} \end{cases}$$

Calculation of $Nu_{L,avg}$ for horizontal, lower surface (Orientation = H, Lower):

$$\bar{Nu}_L = 0.27 Ra_f^{1/4}, \quad 10^5 \leq Ra_f \leq 10^{10}$$

Calculation of $h_{L,avg}$:

$$\bar{h}_L = \bar{Nu}_L \frac{k}{L_c}$$

FIGURE 8.1: Estimated natural convection HTCs along each physically unique surface of the furnace insulation module, assuming $u_s = 440^\circ\text{K}$ and thus $u_s^* = 0.346$.

<p>INPUT VALUE(S)</p> <p>u_s (°K) 610 < Average surface temperature</p> <p>u_∞ (°K) 293 < Ambient air temperature</p> <p>g (m/s²) 9.81 < Gravitational acceleration</p> <p>L [m] 0.40 < Length of surface</p> <p>d [m] 0.60 < Depth of surface</p> <p>Orientation V < Orientation of surface</p>	<p>INTERMEDIATE VALUE(S)</p> <p>L_c (m) 0.40</p> <p>u_f (°K) 451.5 < Film temperature</p> <p>$\beta(u_f)$ (1/°K) 2.21E-03 < Coeff. of volume expansion @ u_f</p> <p>$\nu(u_f)$ (m²/s) 3.21E-05 < Kinematic viscosity of air @ u_f</p> <p>Gr_L (1) 4.28E+08 < Grashof number</p> <p>$Pr(u_f)$ (1) 0.699 < Prandtl number of air @ u_f</p> <p>Ra_L (1) 2.99E+08 < Rayleigh number</p> <p>$Nu_{L,avg}$ (1) 84.7 < Nusselt number (averaged over L)</p> <p>$k(u_f)$ (W/(m°K)) 3.65E-02 < Thermal conductivity of air @ u_f</p>	<p>OUTPUT VALUE(S)</p> <p>$h_{L,avg}$ (W/(m²°K)) 7.73 < HTC (averaged over L)</p>
<p>INPUT VALUE(S)</p> <p>u_s (°K) 610 < Average surface temperature</p> <p>u_∞ (°K) 293 < Ambient air temperature</p> <p>g (m/s²) 9.81 < Gravitational acceleration</p> <p>L [m] 0.40 < Length of surface</p> <p>d [m] 0.60 < Depth of surface</p> <p>Orientation H, Upper < Orientation of surface</p>	<p>INTERMEDIATE VALUE(S)</p> <p>L_c (m) 0.12</p> <p>u_f (°K) 451.5 < Film temperature</p> <p>$\beta(u_f)$ (1/°K) 2.21E-03 < Coeff. of volume expansion @ u_f</p> <p>$\nu(u_f)$ (m²/s) 3.21E-05 < Kinematic viscosity of air @ u_f</p> <p>Gr_L (1) 1.16E+07 < Grashof number</p> <p>$Pr(u_f)$ (1) 0.699 < Prandtl number of air @ u_f</p> <p>Ra_L (1) 8.07E+06 < Rayleigh number</p> <p>$Nu_{L,avg}$ (1) 28.8 < Nusselt number (averaged over L)</p> <p>$k(u_f)$ (W/(m°K)) 3.65E-02 < Thermal conductivity of air @ u_f</p>	<p>OUTPUT VALUE(S)</p> <p>$h_{L,avg}$ (W/(m²°K)) 8.76 < HTC (averaged over L)</p>
<p>INPUT VALUE(S)</p> <p>u_s (°K) 610 < Average surface temperature</p> <p>u_∞ (°K) 293 < Ambient air temperature</p> <p>g (m/s²) 9.81 < Gravitational acceleration</p> <p>L [m] 0.40 < Length of surface</p> <p>d [m] 0.60 < Depth of surface</p> <p>Orientation H, Lower < Orientation of surface</p>	<p>INTERMEDIATE VALUE(S)</p> <p>L_c (m) 0.12</p> <p>u_f (°K) 451.5 < Film temperature</p> <p>$\beta(u_f)$ (1/°K) 2.21E-03 < Coeff. of volume expansion @ u_f</p> <p>$\nu(u_f)$ (m²/s) 3.21E-05 < Kinematic viscosity of air @ u_f</p> <p>Gr_L (1) 1.16E+07 < Grashof number</p> <p>$Pr(u_f)$ (1) 0.699 < Prandtl number of air @ u_f</p> <p>Ra_L (1) 8.07E+06 < Rayleigh number</p> <p>$Nu_{L,avg}$ (1) 14.4 < Nusselt number (averaged over L)</p> <p>$k(u_f)$ (W/(m°K)) 3.65E-02 < Thermal conductivity of air @ u_f</p>	<p>OUTPUT VALUE(S)</p> <p>$h_{L,avg}$ (W/(m²°K)) 4.38 < HTC (averaged over L)</p>

EQUATIONS APPLIED

Calculation of u_f :

$$u_f = \frac{u_s + u_\infty}{2}$$

Calculation of Ra_L :

$$Ra_L = Gr_L Pr = \frac{g \beta (u_s - u_\infty) L_c^3}{\nu^2} Pr$$

Calculation of $Nu_{L,avg}$ for vertical surfaces (Orientation = V)

$$\overline{Nu}_L = \left(0.825 + \frac{0.387 Ra_L^{\frac{1}{4}}}{\left(1 + \left(\frac{0.492}{Pr} \right)^{\frac{4}{3}} \right)^{\frac{1}{4}}} \right)^2$$

Calculation of $Nu_{L,avg}$ for horizontal, upper surface (Orientation = H, Upper):

$$\begin{cases} \overline{Nu}_L = 0.54 Ra_L^{\frac{1}{4}}, & 10^4 \leq Ra_L < 10^7 \\ \overline{Nu}_L = 0.15 Ra_L^{\frac{1}{3}}, & 10^7 \leq Ra_L \leq 10^{11} \end{cases}$$

Calculation of $Nu_{L,avg}$ for horizontal, lower surface (Orientation = H, Lower):

$$\overline{Nu}_L = 0.27 Ra_L^{\frac{1}{4}}, \quad 10^4 \leq Ra_L \leq 10^{10}$$

Calculation of $h_{L,avg}$:

$$\bar{h}_L = \overline{Nu}_L \frac{k}{L_c}$$

FIGURE 8.2: Estimated natural convection HTCs along each physically unique surface of the furnace insulation module, assuming $u_s = 610^\circ\text{K}$ and thus $u_s^* = 0.479$.

9 Appendix II

The complete set of finite difference equations used to compute the numerical solutions presented in Chapter 5 is:

$$\begin{aligned}
\kappa_{i+\frac{1}{2},j}\Delta_i u^* - \kappa_{i-\frac{1}{2},j}\nabla_i u^* + \kappa_{i,j+\frac{1}{2}}\Delta_j u^* - \kappa_{i,j-\frac{1}{2}}\nabla_j u^* &= 0, & i \in (i_\lambda, N-1), j \in (0, j_\lambda) \\
\kappa_{i+\frac{1}{2},j}\Delta_i u^* - \kappa_{i-\frac{1}{2},j}\nabla_i u^* + \kappa_{i,j+\frac{1}{2}}\Delta_j u^* - \kappa_{i,j-\frac{1}{2}}\nabla_j u^* &= 0, & i \in (i_\lambda, N-1), j \in (j_\lambda, N-1) \\
\kappa_{i+\frac{1}{2},j}\Delta_i u^* - \kappa_{i-\frac{1}{2},j}\nabla_i u^* + \kappa_{i,j+\frac{1}{2}}\Delta_j u^* - \kappa_{i,j-\frac{1}{2}}\nabla_j u^* &= 0, & i \in (0, i_\lambda], j \in (j_\lambda, N-1) \\
\frac{\eta}{N-1} \frac{v-u_{ij}^*}{2} - \kappa_{i-\frac{1}{2},j} \frac{\nabla_i u^*}{2} + \kappa_{i,j+\frac{1}{2}} \frac{\Delta_j u^*}{2} - 0 &= 0, & i = N-1, j = 0 \\
\kappa_{i+\frac{1}{2},j} \frac{\Delta_i u^*}{2} - \kappa_{i-\frac{1}{2},j} \frac{\nabla_i u^*}{2} + \kappa_{i,j+\frac{1}{2}} \Delta_j u^* - 0 &= 0, & i \in (i_\lambda, N-1), j = 0 \\
u_{i,j}^* - 1 &= 0, & i = i_\lambda, j \in [0, j_\lambda] \\
u_{i,j}^* - 1 &= 0, & i \in [0, i_\lambda], j = j_\lambda \\
\kappa_{i+\frac{1}{2},j}\Delta_i u^* - 0 + \kappa_{i,j+\frac{1}{2}} \frac{\Delta_j u^*}{2} - \kappa_{i,j-\frac{1}{2}} \frac{\nabla_j u^*}{2} &= 0, & i = 0, j \in (j_\lambda, N-1) \\
\kappa_{i+\frac{1}{2},j} \frac{\Delta_i u^*}{2} - 0 + \frac{\eta}{N-1} \frac{v-u_{ij}^*}{2} - \kappa_{i,j-\frac{1}{2}} \frac{\nabla_j u^*}{2} &= 0, & i = 0, j = N-1 \\
\kappa_{i+\frac{1}{2},j} \frac{\Delta_i u^*}{2} - \kappa_{i-\frac{1}{2},j} \frac{\nabla_i u^*}{2} + \frac{\eta}{N-1} (v - u_{ij}^*) - \kappa_{i,j-\frac{1}{2}} \nabla_j u^* &= 0, & i \in (0, N-1), j = N-1 \\
\frac{\eta}{N-1} \frac{v-u_{ij}^*}{2} - \kappa_{i-\frac{1}{2},j} \frac{\nabla_i u^*}{2} + \frac{\eta}{N-1} \frac{v-u_{ij}^*}{2} - \kappa_{i,j-\frac{1}{2}} \frac{\nabla_j u^*}{2} &= 0, & i = N-1, j = N-1 \\
\frac{\eta}{N-1} (v - u_{ij}^*) - \kappa_{i-\frac{1}{2},j} \nabla_i u^* + \kappa_{i,j+\frac{1}{2}} \frac{\Delta_j u^*}{2} - \kappa_{i,j-\frac{1}{2}} \frac{\nabla_j u^*}{2} &= 0, & i = N-1, j \in (0, N-1),
\end{aligned}$$

where i and j are indexes indicating discrete nodes along the x-axis and y-axis respectively; N is the number of nodes along each axis; $i_\lambda = j_\lambda = \lambda(N-1)$; Δ is the forward difference operator (i.e., $\Delta_i u^* = u_{i+1,j}^* - u_{i,j}^*$); ∇ is the backward difference operator (i.e., $\nabla_i u^* = u_{i,j}^* - u_{i-1,j}^*$);

$$\begin{aligned}
\kappa_{i+\frac{1}{2},j} &= \frac{1}{2} \left(\left(\alpha u_{i+1,j}^{*2} + \beta u_{i+1,j}^* + 1 \right) + \left(\alpha u_{i,j}^{*2} + \beta u_{i,j}^* + 1 \right) \right); \\
\kappa_{i-\frac{1}{2},j} &= \frac{1}{2} \left(\left(\alpha u_{i,j}^{*2} + \beta u_{i,j}^* + 1 \right) + \left(\alpha u_{i-1,j}^{*2} + \beta u_{i-1,j}^* + 1 \right) \right); \\
\kappa_{i,j+\frac{1}{2}} &= \frac{1}{2} \left(\left(\alpha u_{i,j+1}^{*2} + \beta u_{i,j+1}^* + 1 \right) + \left(\alpha u_{i,j}^{*2} + \beta u_{i,j}^* + 1 \right) \right); \text{ and} \\
\kappa_{i,j-\frac{1}{2}} &= \frac{1}{2} \left(\left(\alpha u_{i,j}^{*2} + \beta u_{i,j}^* + 1 \right) + \left(\alpha u_{i,j-1}^{*2} + \beta u_{i,j-1}^* + 1 \right) \right).
\end{aligned}$$

10 Appendix III

The predicted vs. true temperature distribution along each boundary in the problem instances associated with Materials 2-8 are provided in Fig. [10.1-10.7](#). A key for the boundary numbering scheme used is provided in Fig. [6.4](#).

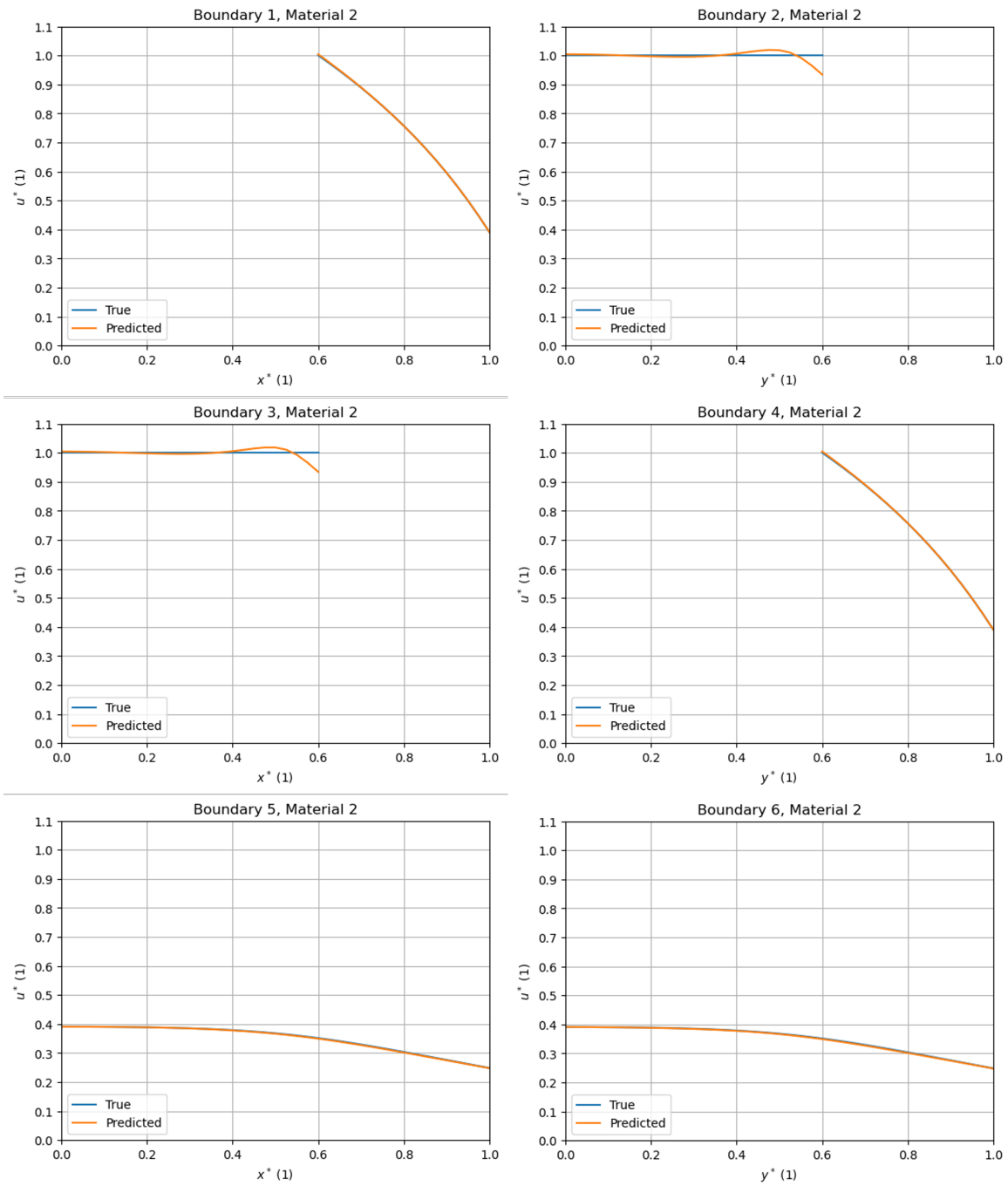


FIGURE 10.1: Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 2.

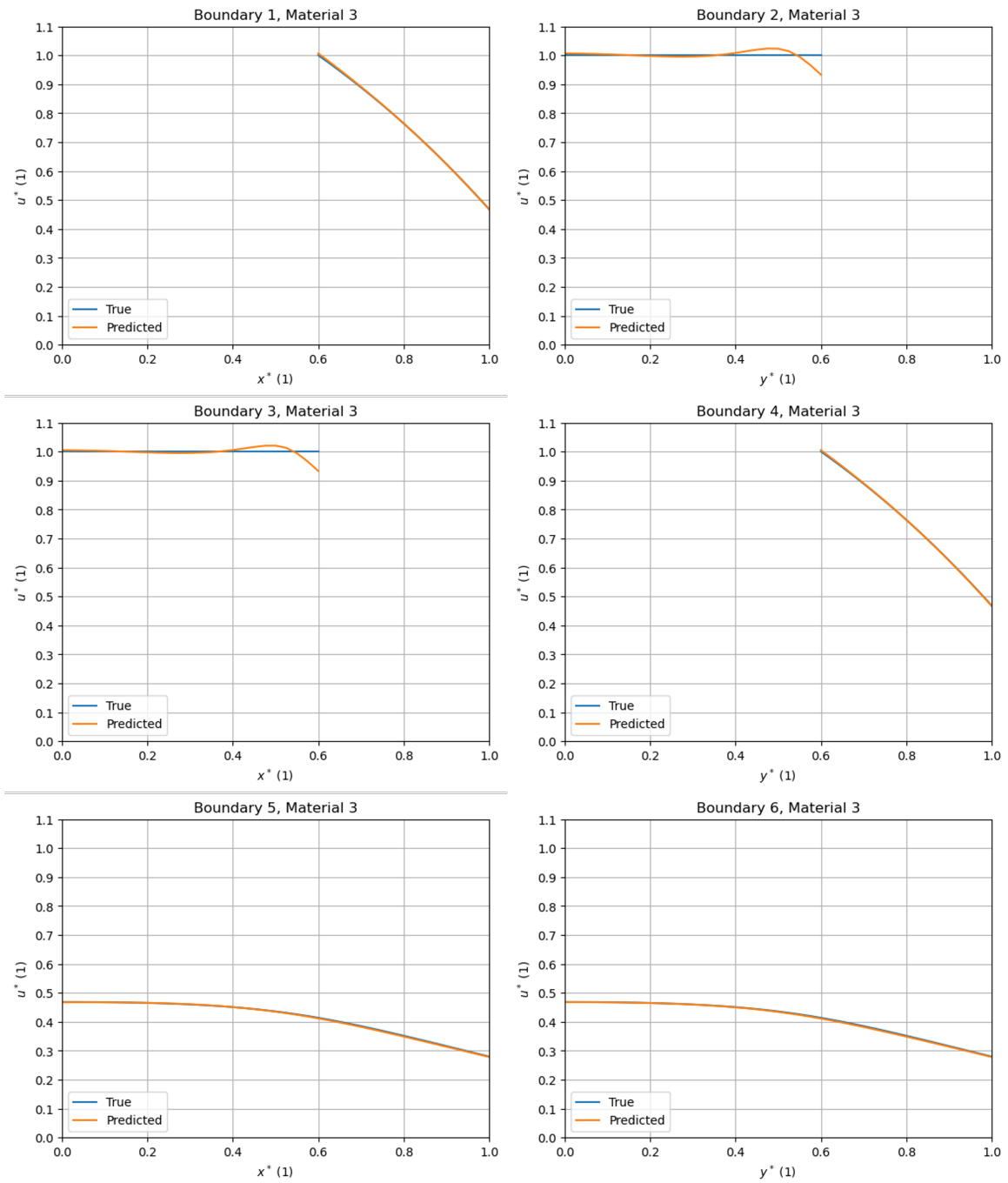


FIGURE 10.2: Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 3.

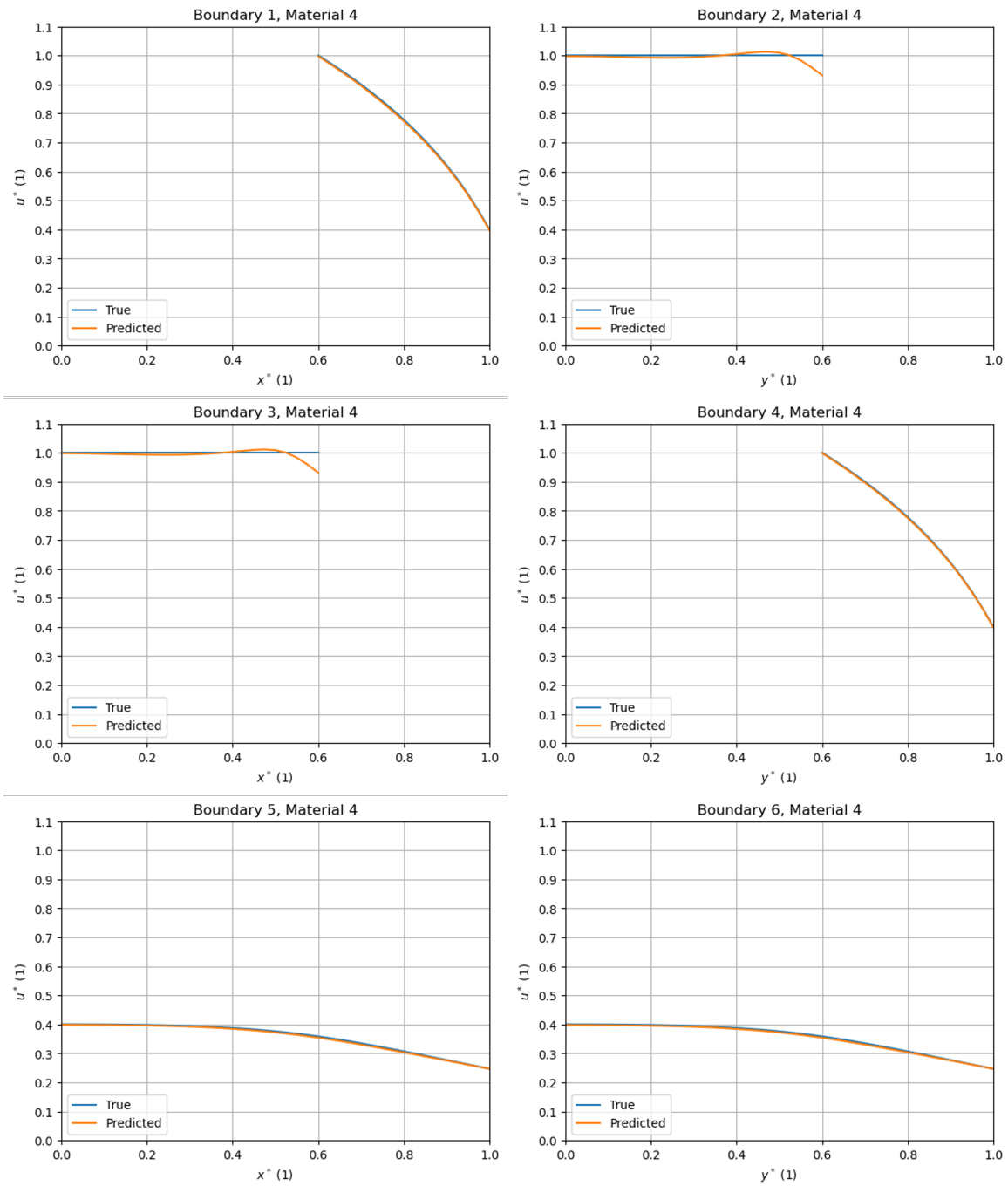


FIGURE 10.3: Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 4.

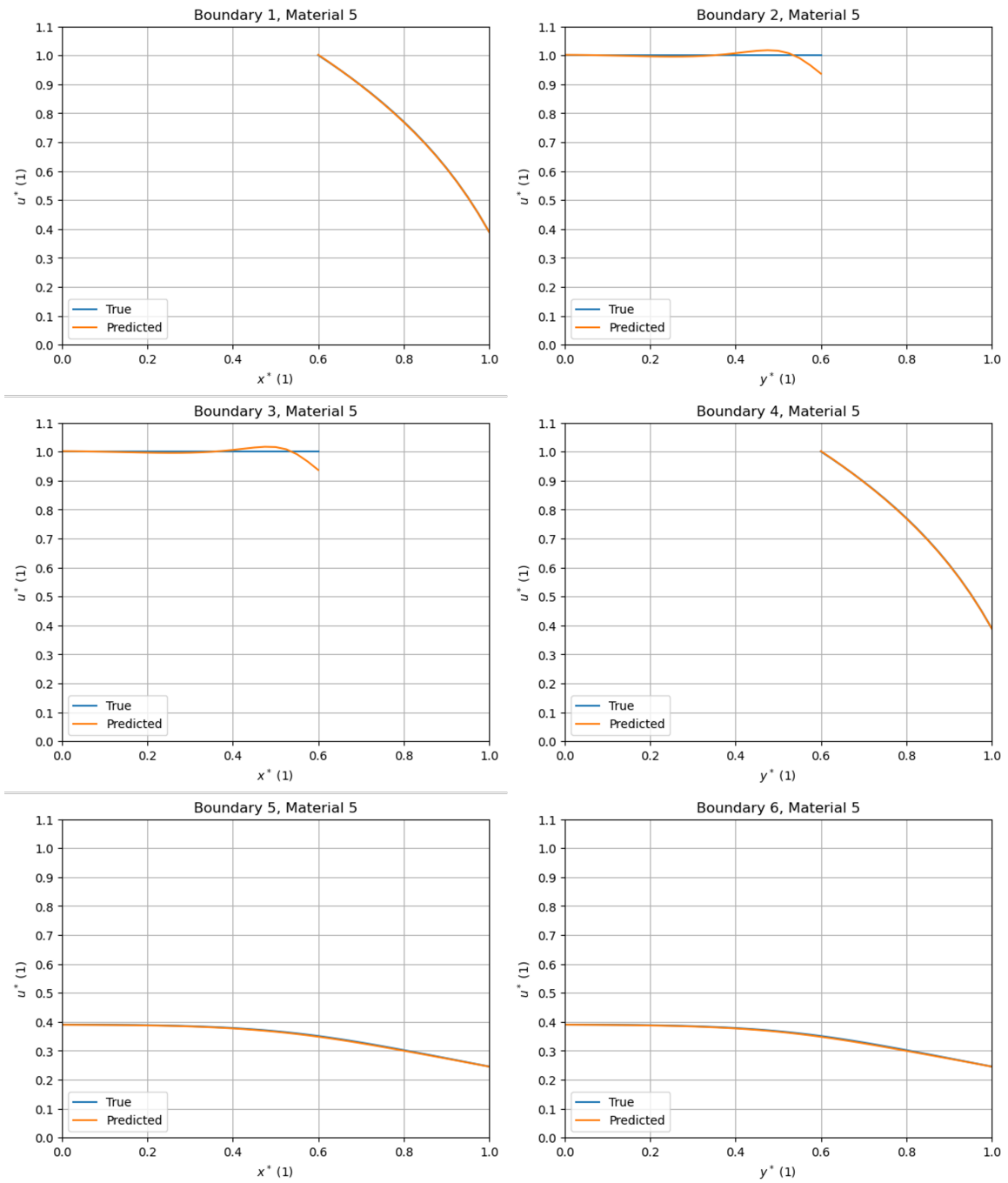


FIGURE 10.4: Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 5.

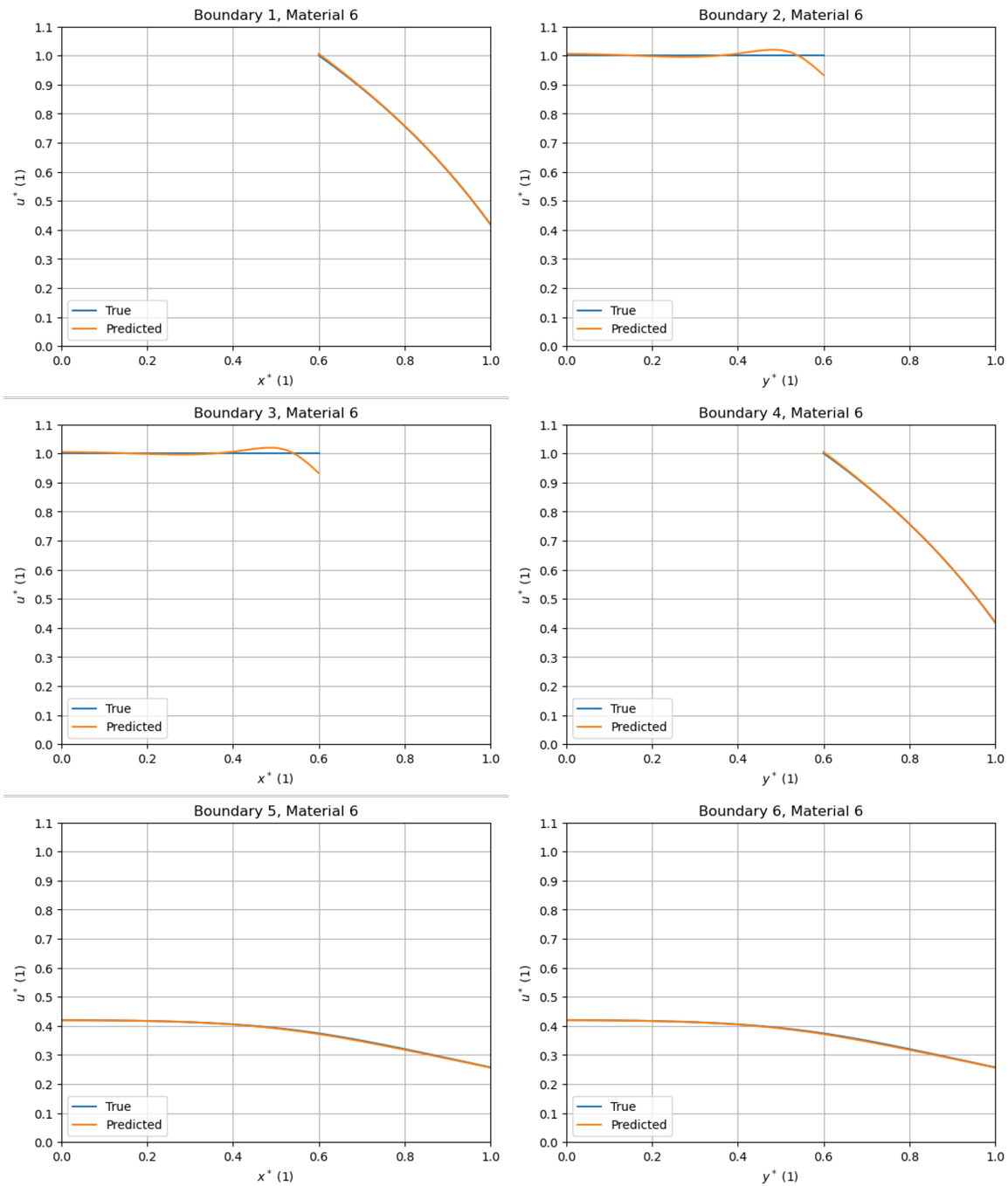


FIGURE 10.5: Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 6.

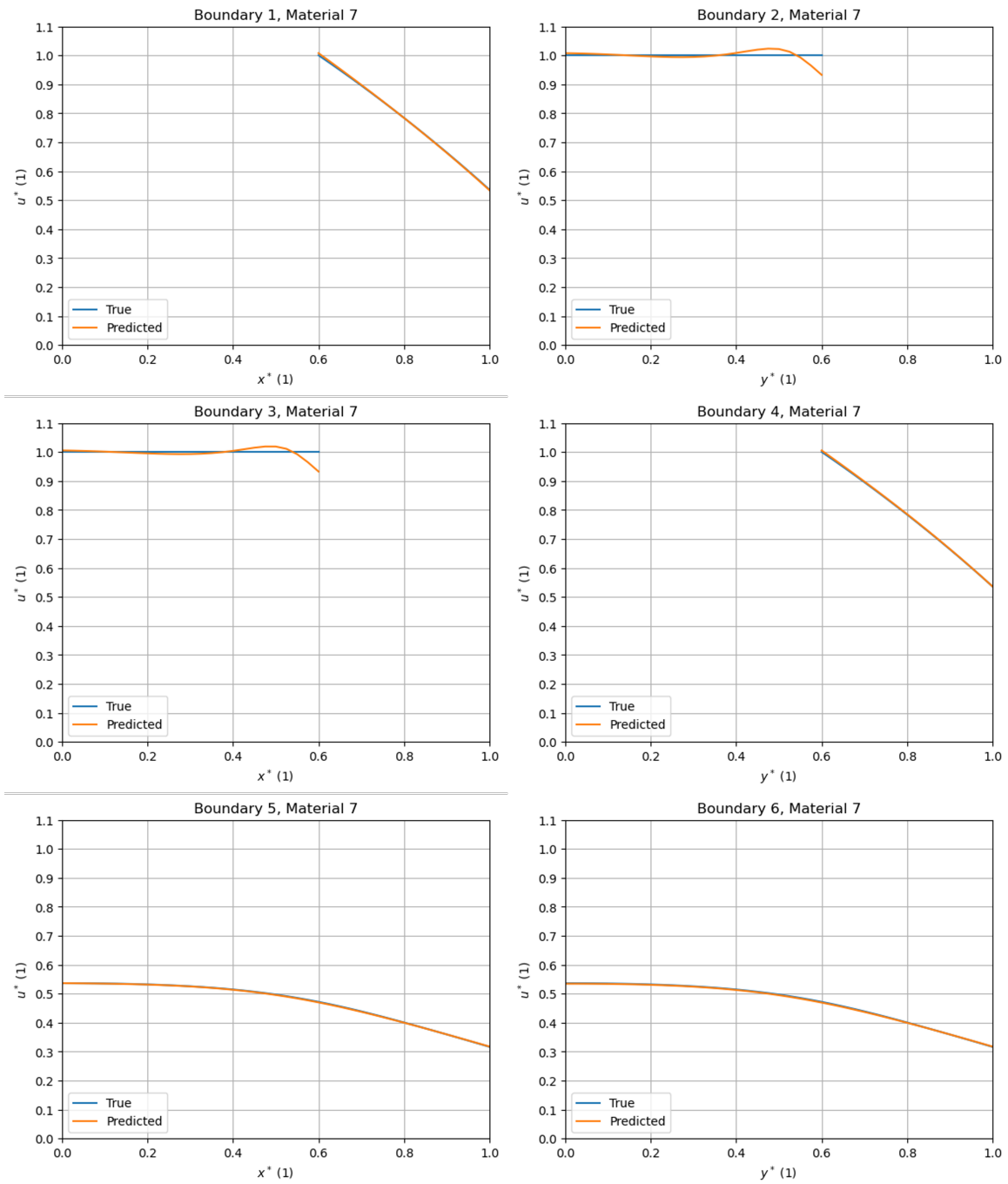


FIGURE 10.6: Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 7.

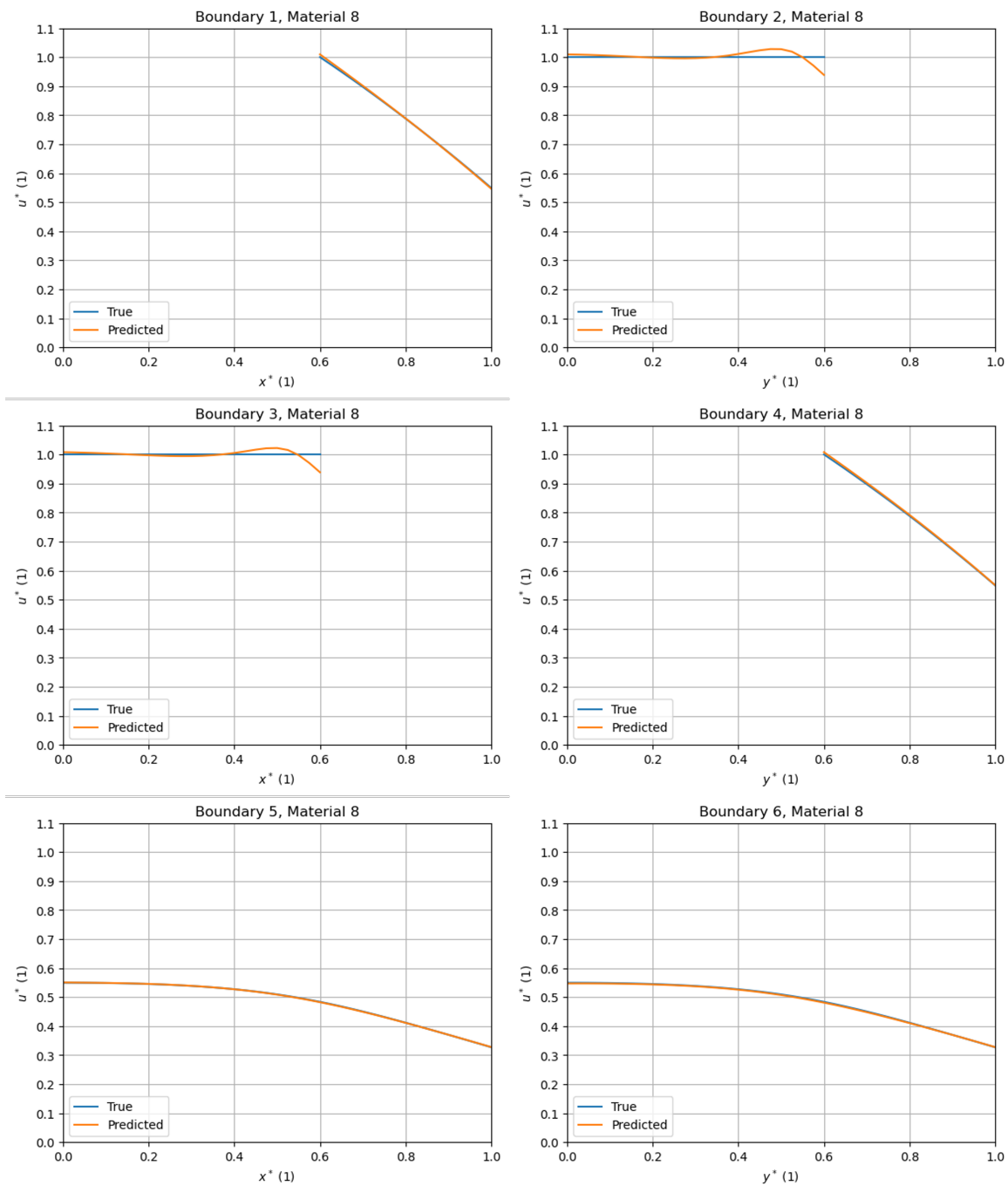


FIGURE 10.7: Comparison of \hat{u}^* and u^* along the domain boundaries in the problem instance associated with Material 8.

Bibliography

- [1] Y. Cengel and A. Ghajar, *Heat and Mass Transfer: Fundamentals & Applications*, 5th ed. New York: McGraw-Hill, 2015.
- [2] F. P. Incropera, D. P. DeWitt, T. L. Bergman, and T. S. Lavine, *Fundamentals of Heat and Mass Transfer*, 6th ed. Hoboken, NJ: John Wiley & Sons, 2007.
- [3] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, pp. 359–366, 1989.
- [4] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, 1989.
- [5] *CUDA C++ Programming Guide, Release 12.1*, NVIDIA, 2023. [Online]. Available: https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.
- [6] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, (Vancouver, Canada), H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.
- [7] M. Abadi, A. Agarwal, P. Barham, *et al.*, *Tensorflow: Large-scale machine learning on heterogeneous systems*, 2015. [Online]. Available: <https://www.tensorflow.org/>.
- [8] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.
- [9] K. Watanabe, I. Matura, M. Abe, M. Kubota, and D. M. Himmelblau, "Incipient fault diagnosis of chemical processes via artificial neural networks," *AIChE Journal*, vol. 35, no. 11, pp. 1803–1812, 1989.
- [10] J. Thibault and B. Grandjean, "A neural network methodology for heat transfer data analysis," *International Journal of Heat and Mass Transfer*, vol. 34, no. 8, pp. 2063–2070, 1991.
- [11] M. W. M. G. Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," *Communications in Numerical Methods in Engineering*, vol. 10, pp. 195–201, 1994.
- [12] K. Jambunathan, S. L. Hartle, S. Ashforth-Frost, and V. N. Fontama, "Evaluating convective heat transfer coefficients using neural networks," *International Journal of Heat and Mass Transfer*, vol. 39, no. 11, pp. 2329–2332, 1996.
- [13] G. Díaz, M. Sen, K. T. Yang, and R. L. McClain, "Simulation of heat exchanger performance by artificial neural networks," *HVACR Research*, vol. 5, no. 3, pp. 195–208, 1999.

- [14] M. Raissi, P. Perdikaris, and G. Karniadakis. "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations." version 0. arXiv: [1711.10561](https://arxiv.org/abs/1711.10561). (2017).
- [15] M. Raissi, P. Perdikaris, and G. Karniadakis. "Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations." version 0. arXiv: [1711.10566](https://arxiv.org/abs/1711.10566). (2017).
- [16] D. Psychogios and L. Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE Journal*, vol. 38, no. 10, pp. 1499–1511, 1992.
- [17] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998.
- [18] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks for heat transfer problems," *Journal of Heat Transfer*, vol. 143, no. 6, 2021.
- [19] O. Hennigh, S. Narasimhan, M. Nabian, *et al.*, "NVIDIA SimNet™: An AI-accelerated multi-physics simulation framework," in *Computational Science - ICCS 2021*, (Krakow, Poland), M. Paszynski, D. Kranzlmuller, V. Krzhizhanovskaya, J. Dongarra, and P. Sloot, Eds., Springer, 2021, pp. 447–461.
- [20] N. Zobeiry and K. Humfeld, "A physics-informed machine learning approach for solving heat transfer equations in advanced manufacturing and engineering applications," *Engineering Applications of Artificial Intelligence*, vol. 101, 2021.
- [21] S. Liao, T. Xue, J. Jeong, S. Webster, K. Ehmann, and J. Cao, "Hybrid thermal modeling of additive manufacturing processes using physics-informed neural networks for temperature prediction and parameter identification," *Computational Mechanics*, vol. 72, pp. 499–512, 2023.
- [22] T. Wurth, C. Krauss, C. Zimmerling, and L. Karger, "Physics-informed neural networks for data-free surrogate modelling and engineering optimization - An example from composite manufacturing," *Materials Design*, vol. 231, 2023.
- [23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [24] A. Baydin, B. Pearlmutter, A. Radul, and J. Siskind, "Automatic differentiation in machine learning: A survey," *Journal of Machine Learning Research*, vol. 18, 2018.
- [25] L. Sun, H. Gao, S. Pan, and J. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," *Computer Methods in Applied Mechanics and Engineering*, vol. 361, 2020.
- [26] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM Journal on Scientific Computing*, vol. 43, 2021.
- [27] *Laboratory Industrial Ovens Furnaces*, Carbolite-Gero. [Online]. Available: <https://www.carbolite-gero.com/files/12830/laboratory-industrial-ovens-furnaces.pdf>.

- [28] *Kaowool® Organic Boards*, Morgan Advanced Materials, 2021. [Online]. Available: https://www.morganadvancedmaterials.com/media/4xtlvoq0/kaowool-organic-boards-global_eng.pdf.
- [29] *Inorganic Boards*, Morgan Advanced Materials, 2020. [Online]. Available: https://www.morganadvancedmaterials.com/media/mccfjp2d/inorganic-i-boards_eng.pdf.
- [30] *Alumina Type AL-30*, ZIRCAR Ceramics. [Online]. Available: <https://zircarceramics.com/wp-content/uploads/2017/01/AL-30.pdf>.
- [31] *Alumina Type SALI*, ZIRCAR Ceramics. [Online]. Available: <https://zircarceramics.com/wp-content/uploads/2017/02/SALI.pdf>.
- [32] *Alumina Type SALI-2*, ZIRCAR Ceramics. [Online]. Available: <https://zircarceramics.com/wp-content/uploads/2017/01/SALI-2.pdf>.
- [33] *Consistent Performance at a High Degree: Thermo Scientific Furnaces*, Thermo Fisher Scientific, 2023. [Online]. Available: <https://assets.thermofisher.com/TFS-Assets%2FLED%2Fbrochures%2FLED-FurnacesBrochure-BRFURNACE0316-EN.pdf>.
- [34] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.
- [35] Y. Cengel and A. J. Cimbala, *Fluid Mechanics: Fundamentals and Applications*, 1st ed. New York: McGraw-Hill, 2006.
- [36] J. Leinhard and J. Leinhard, *A Heat Transfer Textbook*, 4th ed. Cambridge, MA: Phlogiston Press, 2017.
- [37] E. Buckingham, “On physically similar systems: Illustrations of the use of dimensional equations,” *Physical Review*, vol. 4, no. 4, pp. 345–376, 2018.
- [38] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, “SciPy 1.0: Fundamental algorithms for scientific computing in python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [39] Z. Xiang, W. Peng, X. Liu, and W. Yao, “Self-adaptive loss balanced physics-informed neural networks,” *Neurocomputing*, vol. 496, pp. 11–34, 2022.
- [40] R. van der Meer, C. W. Oosterlee, and A. Borovykh, “Optimally weighted loss functions for solving PDEs with neural networks,” *Journal of Computational and Applied Mathematics*, vol. 405, 2022.
- [41] D. Kingma and J. Ba. “Adam: A method for stochastic optimization.” version 9. arXiv: [1412.6980](https://arxiv.org/abs/1412.6980). (2017).
- [42] C. N. Kroll and J. R. Stedinger, “Estimation of moments and quantiles using censored data,” *Water Resources Research*, vol. 32, no. 4, pp. 1005–1012, 1996.
- [43] S. P. Wechsler and C. N. Kroll, “Quantifying DEM uncertainty and its effect on topographic parameters,” *Photogrammetric Engineering Remote Sensing*, vol. 72, no. 9, pp. 1081–1090, 2006.
- [44] M. D. McKay, R. J. Beckman, and W. J. Conover, “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” *Technometrics*, vol. 42, no. 1, pp. 55–61, 1979.