

A digital twin of fusion energy components with physics-informed neural networks

Prakhar Sharma

*submitted to Swansea University in fulfilment of
the requirements for the Degree of
Doctor of Philosophy*



SWANSEA UNIVERSITY
SEPTEMBER 2024

Copyright: The Author, Prakhar Sharma, 2024

Distributed under the terms of a Creative Commons Attribution-NonCommercial-NoDerivatives
4.0 International License (CC BY-NC-ND 4.0).

Abstract

A fusion reactor presents a highly challenging operational environment, with engineering components exposed to extreme conditions. Temperatures range from 100 million °C at the centre of the plasma to -269 °C in the cryopump, all within a few metres. In addition to these temperature extremes, components must withstand intense electromagnetic loads and irradiation damage. These extreme conditions were achieved for short periods at JET, formerly the world's largest fusion device located at Culham Centre for Fusion Energy, UK, before it ceased operations. But one of the greatest engineering challenges of the 21st century will be to construct a machine that can operate under these extremes routinely and produce commercially viable energy.

To create a fusion reactor, relevant components must undergo rigorous testing before they can be deployed. Initially, testing at the HIVE facility focused on the sample under test (SUT) under thermal loads alone. However, with the development of testing facilities such as CHIMERA, it will soon be possible to test the SUT under both thermal and magnetic loads. Given the challenges associated with testing, any additional information to understand these components would be extremely useful. This project investigates the potential for creating a foundational digital twin capable of replicating reality from sparse data inputs. A digital twin enables real-time monitoring, provides predictive insights, and supports decision-making to optimise experimental campaigns and ensure safety.

In this context, the thesis investigates the suitability of physics-informed neural networks (PINNs) for solving inverse problems and extends their application to develop a foundational physics-informed digital twin platform for replicating physical experiments. PINNs are ideal for scenarios with limited data availability, as they do not require extensive pre-training. They provide a straightforward approach to solving inverse problems without complex algorithms, making them ideal for enhancing testing processes in experimental facilities.

This study demonstrates that PINNs can accurately reconstruct thermal fields from sparse data and solve inverse problems with challenging boundary conditions (BCs). The developed PINN-based digital twin reconstructed steady-state temperature distributions of the HIVE sample under varying thermal loads, enabling detailed analysis of the temperature field and the corresponding thermal conductivity variation. These findings establish PINNs as a promising tool for inverse modelling with limited data.

Declarations

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.




Prakhar Sharma
24th December 2024

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.



Prakhar Sharma
24th December 2024

I hereby give consent for my thesis, if accepted, to be available for electronic sharing.



Prakhar Sharma
24th December 2024

The University's ethical procedures have been followed and, where appropriate, that ethical approval has been granted.



Prakhar Sharma
24th December 2024

Acknowledgements

I am immensely thankful to my main supervisor, Prof. Perumal Nithiarasu, for his unparalleled guidance, unwavering support, and patience throughout my PhD journey. His profound expertise not only shaped the direction of my research but also opened doors to numerous opportunities, such as invitations to contribute to special issues and book chapters. Prof. Nithiarasu's mentorship has been a cornerstone of my academic and professional development. Equally, I extend a deep sense of gratitude to Dr. Llion Evans for his instrumental role in shaping my PhD experience. He was not only integral in involving me in the development of 'VirtualLab' project but also provided unparalleled support with logistical tasks, document preparation, and crafting presentations for crucial meetings. His guidance was invaluable during his tenure at the university. I would also like to thank my industrial supervisor, Michelle Tindall, for her invaluable support throughout the PhD and introducing me to wonderful people at the UKAEA.

I would like to express my sincere gratitude to Supercomputing Wales for granting me access to the Sunbird HPC cluster and Cardiff University's Hawk HPC cluster. I would also like to thank the University of Edinburgh for granting me access to ARCHER2. Additionally, I owe a great deal of gratitude to Rhydian Lewis, who not only shared his expertise in simulations, similar to those he developed during his PhD, but also helped me to deeply understand his works. Furthermore, I learned a great deal about software development from both Ben Thorpe and Rhydian Lewis during the development of 'VirtualLab.' Their insights later enabled me to create a sophisticated library of my own. My gratitude also extends to the Ed Bennett, Tom Pritchard and Tianyi Pan for their continuous support with the Sunbird. Their expertise and assistance were vital in harnessing the full potential of HPC resources for my research.

I would like to express my sincere gratitude to EPSRC and UKAEA for their generous sponsorship of my PhD program. I am also grateful to NVIDIA for awarding us the NVIDIA academic hardware grant. Lastly, my heartfelt appreciation goes out to my family for their unwavering encouragement and support throughout my studies.

Contents

Abstract	I
Declarations	II
Acknowledgements	III
List of Figures	VIII
List of Tables	XIII
List of Algorithms	XIV
List of Abbreviations	XV
1 Introduction	1
1.1 Fusion energy	1
1.2 HIVE experimental facility	2
1.3 CHIMERA experimental facility	4
1.4 Testing of fusion components	4
1.5 Inverse problem and digital twin	5
1.6 Aims and Objectives	6
1.7 Outline of the thesis	7
1.8 Dissemination of research	8
1.8.1 Research papers	9
1.8.2 Conference presentations	9
2 Physics-informed Neural Networks	10
2.1 Introduction	10
2.2 Physics-driven vs. Data-driven approaches	11

2.3	How PINNs are different from conventional numerical techniques?	13
2.4	Neural networks for regression	13
2.4.1	Forward pass	15
2.4.2	Backpropagation and automatic gradient	16
2.4.3	Complexity of analytical derivatives in neural networks	18
2.5	The baseline PINN	19
2.5.1	Discrete loss function	20
2.5.2	Reconstruction loss	21
2.5.3	Integral formulation of loss	22
2.5.4	Issues with the baseline PINN	23
2.5.5	Nature of solutions in PINNs	23
2.5.6	Loss metrics	24
2.6	Developments in the PINN frameworks	24
2.6.1	Sampling	25
2.6.2	Imbalanced loss terms	26
2.6.3	Activation function	27
2.6.4	Spectral bias in the NN	28
2.7	Applications of PINNs to forward problems	28
2.7.1	Parametric problems	28
2.7.2	Complex geometry	29
2.7.3	Transfer learning	30
2.8	Modified baseline PINNs	30
2.8.1	Fourier Network	31
2.8.2	Modified Fourier neural network	32
2.8.3	Sinusoidal Representation Networks (SiReNs)	33
2.9	Examples	34
2.9.1	1D Burgers equation	34
2.9.2	1D Allen-Cahn equation	36
2.9.3	Lid driven cavity	37
2.10	Conclusions	41
3	Solving stiff-PDEs using PINNs	43
3.1	Challenges in Solving Stiff-PDEs with PINNs	44
3.1.1	Exact BC Imposition	44
3.1.2	Overfitted vs generalised solution	44

3.2	Tools for Solving Stiff-PDEs with PINNs	45
3.2.1	Signed distance function	45
3.2.2	Importance sampling	45
3.3	Open-source libraries	46
3.3.1	DeepXDE	46
3.3.2	NVIDIA Modulus	47
3.4	Problem 1: 2D steady-state heat conduction	48
3.4.1	Model 1: baseline PINN	49
3.4.2	Model 2: DeepXDE's baseline PINN	56
3.4.3	Models 3-11: NVIDIA Modulus	56
3.4.4	Hard constrained BCs	59
3.4.5	Overfitted solution	60
3.5	Problem 2: 3D steady-state heat conduction	60
3.6	Parametric heat conduction problem	68
3.6.1	Problem 3: Parameterised conductivity	69
3.6.2	Problem 4: Parameterised geometry	69
3.7	Conclusion	74
4	Hyperparameter tuning of PINNs	78
4.1	Introduction	78
4.2	Summary of methodology	79
4.3	Test cases	81
4.3.1	Problem 1 : 2D steady-state heat conduction	81
4.3.2	Problem 2 : 3D steady-state heat conduction	81
4.3.3	Problem 3 : Problem 1 with parametric geometry	82
4.3.4	Problem 4 : Problem 1 with parametric conductivity	84
4.3.5	Problem 5 : 2D steady-state heat conduction with discontinuous conductivity	84
4.4	Results and discussion	84
4.5	Conclusions	90
5	Inverse solution reconstruction	92
5.1	Introduction	92
5.2	The proposed workflow	94
5.2.1	Inverse workflow	95
5.2.2	Forward workflow	101

5.3	Results and discussion	102
5.3.1	Problem 1 : 2D steady-state heat conduction with Neumann BC	103
5.3.2	Problem 2 : 2D steady-state heat conduction with a Sine BC	105
5.3.3	Problem 3 : 2D Helmholtz problem over a square domain	107
5.4	Conclusion	109
6	HIVE digital twin	111
6.1	HIVE experiment	111
6.1.1	Induction heating	112
6.1.2	Active cooling	113
6.2	The AMAZE sample	113
6.3	Numerical simulation with VirtualLab	114
6.4	Physics-informed digital twin	116
6.4.1	Assumptions	116
6.4.2	Revisiting the PINN-based workflow	117
6.5	Results and discussion	118
6.5.1	Initial simulations to identify suitable hyperparameters	118
6.5.2	Transfer learning	119
6.6	Conclusion	121
7	Conclusion and future work	123
7.1	Conclusions	123
7.1.1	Advantages of PINNs	123
7.1.2	Disadvantages of PINNs	124
7.1.3	Solution reconstruction workflow	125
7.1.4	Digital twinning	125
7.2	Future work	126
7.2.1	Better diagnostics	126
7.2.2	Reconstruction of complicated simulations	126
7.2.3	Faster reconstruction	126
7.2.4	Other machine learning techniques	127

List of Figures

1.1	A comparison of the plasma volumes in JET, JT-60SA, ITER and DEMO.	2
1.2	Photograph of the HIVE experimental setup taken in the lab.	3
2.1	A broad classification of different approaches for solving physics problems.	12
2.2	Schematic diagram of an artificial neuron. Here, $\mathbf{X} = \{1, x_1, x_2, \dots x_m\}$ is the input dataset, $\mathbf{W} = \{w_0, w_1, w_2, \dots w_m\}$ is the vector of trainable weights/ parameters, σ is the activation function and \hat{y} is the predicted output.	15
2.3	Schematic diagram of a fully connected neural network (FCNN) with 2 hidden layers. Here, $\mathbf{X} = \{1, x_1, x_2, \dots x_m\}$ is the input dataset, $\{w^{(1)}, w^{(2)}, w^{(3)}\}$ is the vector of trainable weights/ parameters for each layer, $\{a^{(2)}, a^{(3)}\}$ is the vector of activated layers and \hat{y} is the predicted output.	16
2.4	The architecture of a PINN with two hidden layers for a 3D spatio-temporal forward problem. Here the inputs are x, y, z and t , σ is the activated neuron and \hat{u} is the predicted output or the solution to the PDE.	21
2.5	The comparison of various sampling methods.	25
2.6	The comparison of initial and importance sampling strategies in a 1D feature space x . The top histogram shows initial samples obtained via LHS, and the target distribution of samples derived from importance sampling. The bottom plot shows the pointwise total loss \mathcal{L} across the 1D feature space x , highlighting regions of higher loss where more points are sampled.	26
2.7	Histogram of the magnitude spectrum obtained from the Fourier transform of 2D spatial data, indicating a the presence of high-frequency components due to discontinuities.	28
2.8	Schematic representation of the training dataset with varying parameter k	29

2.9	The architecture of a PINN a 3D spatio-temporal forward problem with a varying parameter k . Here the inputs are $X = x, y, z, t$ and k , \hat{u} is the predicted output or the solution to the PDE.	30
2.10	Transfer learning from a base task to a target task with different geometry and BC, indicated by '*'.	31
2.11	Structure of modified Fourier network as per Equation 2.20.	32
2.12	Structure of DGM architecture and the DGM layers as per Equation 2.21.	35
2.13	Baseline PINN predicted solution of 1D Burgers equation is shown on the top and absolute pointwise error between the analytical solution and PINN predicted solution is shown on the bottom.	36
2.14	Comparison of Baseline PINN and Fourier NN predicted solutions for the 1D Allen-Cahn equation. The top row of each subfigure shows the predicted solutions, while the bottom row displays the absolute pointwise error relative to the analytical solution.	38
2.15	Geometry of the lid-driven cavity problem.	39
2.16	Numerical solution for the lid-driven cavity problem, obtained using the SIMPLE algorithm, showcasing the velocity field and pressure distribution.	40
2.17	Baseline PINN predicted solution of the lid-driven cavity.	40
2.18	DGM predicted solution of the lid-driven cavity.	40
2.19	DGM predicted solution of the lid-driven cavity with three sub-domains equally spaced along the x -direction.	41
3.1	FEM solution of 2D steady-state heat conduction problem (Equation 3.4) using MATLAB Partial Differential Equation Toolbox.	48
3.2	Solution of 2D steady-state heat conduction problem using Model 1. The predicted temperature distribution (on the top) and the absolute pointwise error (on the bottom) is shown at 5k, 10k and 30k epochs respectively. Temperature predicted outside the expected bound i.e. when $u \notin [0, 1]$, is shown using the white and grey colour.	49
3.3	The training loss (top) and the validation loss (bottom) for the 2D steady-state heat conduction problem. The training loss plot shows the BC loss (BC loss), the residual loss (PDE loss) and the sum of both, i.e. the total loss. The validation loss plot shows the mean squared error between the temperature predicted from the training dataset and the FEM solution. The black dot denotes the least validation loss.	51

3.4	PINN predicted solution of 2D steady-state heat conduction problem is shown on the left side. The absolute pointwise error between the FEM solution and PINN predicted solution is shown on the right side. Temperature predicted outside the expected bound (if any), i.e. when $u \notin [0, 1]$ is shown using the white and grey colour.	55
3.5	The magnitude of SDF weights on interior and boundary points of a square domain. The SDF weights for Model 4, i.e. with only interior points is shown on the left side. Whereas, the SDF weights with interior and boundary points is shown on the right side (for Model 5 to 11).	57
3.6	DeepXDE predicted solution of 2D steady-state heat conduction problem (Equation 3.4) with hard constrained BCs at the top and bottom walls on the left and the absolute pointwise error between the FEM and PINN predicted solutions is shown on the right side. Temperature predicted outside the expected bound (if any), i.e. when $u \notin [0, 1]$ is shown using the white and grey colour.	59
3.7	Comparison of PINN predicted solution from the training dataset and at new data-points. (a) The FEM solution of Problem 1 (Equation 3.4), (b) The DeepXDE predicted solution of the same problem from the training dataset, and (c) the DeepXDE inferred solution at new locations in the domain using the pretrained PINN from (b). Temperature predicted outside the expected bound (if any), i.e. when $u \notin [0, 1]$ is shown using the white and grey colour.	60
3.8	FEM solution for the 3D steady-state heat conduction problem (Equation 3.7) using MATLAB Partial Differential Equation Toolbox.	61
3.9	PINN predicted solution of 3D steady-state heat conduction problem is shown on the left side. The absolute pointwise error between the FEM solution and PINN predicted solution is shown on the right side. Temperature predicted outside the expected bound (if any), i.e. when $u \notin [0, 1]$ is shown using the white and grey colour.	66
3.10	The magnitude of SDF weights on interior points of the cubical domain of problem 2.	67
3.11	Solution to Problem 3 using Model 2. The predicted temperature distribution (on the top) and the absolute pointwise error (on the bottom) for different values of κ (see Equation 3.8). Temperature predicted outside the expected bound, i.e. when $u \notin [0, 1]$, is shown in white and grey colour.	70
3.12	The boundary points sampling of the parameterised geometry in Problem 4. . . .	71

3.13	Solution of 2D steady-state heat conduction problem with parameterised upper boundary. Each plot shows the predicted temperature distribution for a specific value of parameter $L \in [1, 1.05]$. Temperature predicted outside the expected bound, i.e. when $u \notin [0, 1]$, is shown using white and grey colour.	74
4.1	FEM Solution of Problem 1 (left) and Problem 2 (right).	82
4.2	The loss curve (training loss) of Problem 1 trained with Tanh activation and baseline PINN for 14k iterations.	83
4.3	The pointwise BC loss (left) and pointwise PDE loss (right) of Problem 1 trained with Tanh activation and baseline PINN for 14k iterations.	83
4.4	Comparison of activation function for five different test cases. The horizontal axis shows the activation function while the vertical axis shows the mean relative L2 error.	86
4.5	Comparison of number of hidden layers for five different test cases. The horizontal axis shows the number of hidden layers while the vertical axis shows the mean relative L2 error. For each specific number of hidden layers, the mean relative L2 error includes data from all architectures with that specific number of hidden layers.	87
4.6	Comparison of neural network architectures for five different test cases. The horizontal axis shows the individual neural network architectures while the vertical axis shows the mean relative L2 error.	88
4.7	Comparison of learning rates for five different test cases. The horizontal axis shows the learning rates while the vertical axis shows the mean relative L2 error.	89
4.8	The distribution of relative L2 error with optimised settings across all the test cases.	91
5.1	Schematic diagram of the proposed workflow. The inverse workflow is on the left and the forward workflow is on the right.	95
5.2	The ReLU function.	97
5.3	The FEM solution (ground truth) of Problem 1 showing the field variable $u(x, y)$ on the left. The location of 4 boundary and 2 interior sparse measurements and the point cloud used for the inverse workflow are shown on the right.	104
5.4	The variation of the coefficients of Problem 1 with training iterations. The sparsity knob is shown in dashed grey colour. Any coefficient which remains below the sparsity knob for 2000 consecutive iterations was excluded from further training.	105

5.5	The left side of the figure presents the FEM solution (ground truth) for Problem 2 showing the field variable $u(x, y)$. The location of 4 boundary and 2 interior sparse measurements and the point cloud used for the inverse workflow are shown on the right.	106
5.6	Sensitivity of the relative L2 error to white noise across different orders of $\mathcal{G}(x)$ in Problem 2.	107
5.7	The FEM solution (ground truth) of Problem 3 showing the field variable $u(x, y)$ on the left. The location of 16 boundary and 10 interior sparse measurements and the point cloud used for the inverse workflow are shown on the right.	108
5.8	A flow chart of the transfer learning approach used in Problem 3.. . . .	109
6.1	A generic HIVE sample showing its individual components.	112
6.2	The AMAZE sample with the coil setup.	113
6.3	Drawing of the AMAZE sample. All dimensions are shown in meters.	114
6.4	The 1D boiling curve used in HIVE. The curve illustrates different boiling regimes, marked by key points: saturation temperature (blue 'x'), onset of nucleate boiling (orange 'x'), and critical heat flux (green 'x'). Heat flux increases non-linearly with temperature, peaking at the critical heat flux before transitioning to film boiling.	115
6.5	The PINN-based workflow for solution reconstruction.	117
6.6	The temperature profile from the thermo-mechanical simulation of the AMAZE sample.	118
6.7	Manually sampled 10 sparse measurements are shown on the visible surfaces of the AMAZE sample. The remaining 5 sparse measurements, not visible in this figure, are mirror images on the opposite, invisible side.	120

List of Tables

2.1	Conventional numerical techniques vs PINNs	13
2.2	Relative L2 error (in %) for Lid driven cavity.	41
3.1	Models considered in the numerical study	50
3.2	Problem 1: Summary of training parameters and relative L^2 error for different PINN frameworks.	50
3.3	Problem 2: Summary of training parameters and relative L^2 error for different PINN frameworks	62
3.4	Summary of the total training time (sec) and training time (sec) per epoch in Problem 1 and 2 for each model.	68
3.5	Summary of parametric heat conduction problems	69
3.6	Summary of models with least relative L^2 error.	76
4.1	Range of hyperparameters explored.	80
4.2	Mesh statistics of the FEM solution.	82
4.3	Optimal hyperparameter settings.	90
5.1	The relative L2 error (%) and the maximum absolute error for each problem with different levels of white noise. For every problem, the first row denotes the relative L2 error, and the second row indicates the maximum absolute error.	103
6.1	Comparison of relative L2 errors (in %) and average time taken (min) for different currents (A) using independent steady-states and transfer learning.	120
6.2	The variation in the minimum thermal conductivity minimum with increasing input current, calculated using the hypothetical relationship between thermal conductivity and temperature.	121

List of Algorithms

5.1 Promoting sparsity in polynomial function (Equation 5.11)	100
---	-----

List of Abbreviations

Abbreviations

AC	Alternating current
BC	Boundary condition
CAD	Computer aided design
CFD	Computational fluid dynamics
CHIMERA	Combined heating and magnetic research apparatus
DEMO	DEMOstration power plant
DGM	Deep Galerkin method
DNN	Deep neural network
EM	Electromagnetic
FCNN	Fully connected neural network
FDM	Finite difference method
FEM	Finite element method
FVM	Finite volume method
GPR	Gaussian process regression
HHF	High heat flux
HIVE	Heat by induction to verify extremes
HPC	High performance computing
IC	Initial condition
IHTP	Inverse heat transfer problem
ITER	International thermonuclear experimental reactor
JET	Joint European torus
JH	Joule heating
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm
LSTM	Long short-term memory

MAST	Mega ampere spherical Tokamak
ML	Machine learning
MLP	Multi layer perceptron
MSE	Mean squared error
ODE	Ordinary differential equation
PDE	Partial differential equation
PFC	Plasma facing component
PINN	Physics-informed neural network
RAM	Random access memory
RNN	Recurrent neural network
SciML	Scientific machine learning
SDF	Signed distance function
SiReNs	Sinusoidal representation network
STEP	Spherical Tokamak for energy production
SUT	Sample under test
UKAEA	United Kingdom atomic energy authority

Chapter 1

Introduction

1.1 Fusion energy

The world's energy consumption has increased sevenfold in past 80 years due to technological advancements. Historically, the primary sources of energy production have been fossil fuels such as coal, oil, and natural gas. The burning of fossil fuels releases greenhouse gases such as carbon dioxide (CO_2) and methane (CH_4), resulting in rise in global atmospheric temperatures, often referred to as global warming. In the last 2-3 decades, many countries have committed to achieving net-zero greenhouse gas emissions by 2050 [1, 2]. While renewable energy are integral to the concept of a net-zero energy portfolio, their supply is unreliable due to weather dependence. Therefore, it is essential to have other energy sources available simultaneously.

Fusion energy, frequently referred to as the 'holy grail' of energy sources, holds significant potential for the future energy mix. Unlike fission, nuclear fusion combines lighter nuclei to create a heavier nucleus, releasing three to four times more energy than nuclear fission per unit of fuel. Additionally, fusion does not emit CO_2 or other harmful greenhouse gases, thereby not contributing to global warming.

To achieve nuclear fusion on Earth, plasma must be heated to over 150 million $^{\circ}\text{C}$, approximately ten times hotter than the Sun's core. This extreme condition was consistently achieved at the Joint European Torus (JET) fusion facility, located at the UK Atomic Energy Authority (UKAEA) site in Culham, over the past 40 years. JET, once the world's largest and most advanced operational tokamak, was not capable of producing net energy, with its fusion energy gain factor (Q) was 0.6. As of December 2023, JET has ceased operations, with its decommissioning planned to continue until 2040 [3]. Currently, the most powerful tokamak is JT-60SA in Japan. The international thermonuclear experimental reactor (ITER) in France, still under construction, and

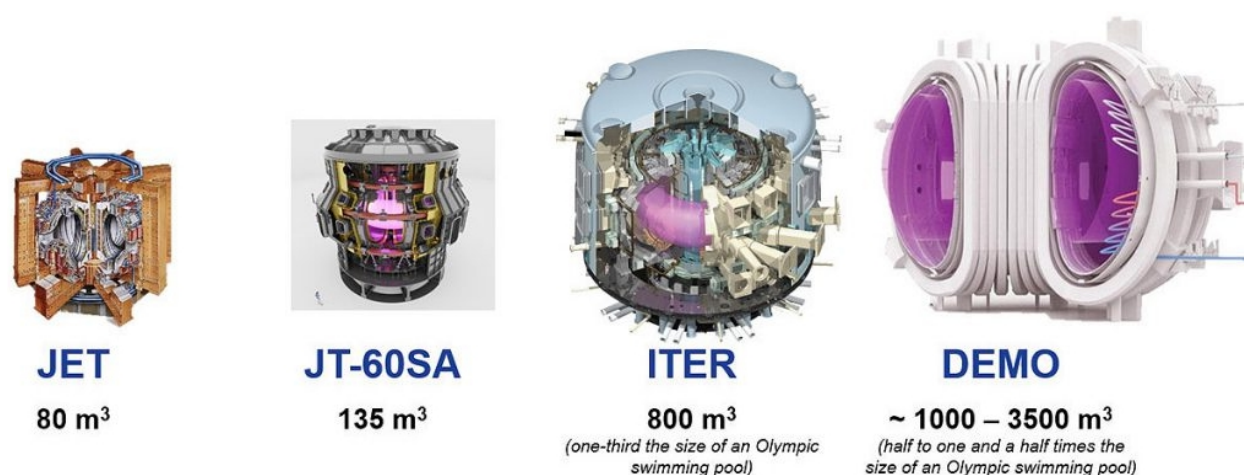


Figure 1.1: A comparison of the plasma volumes in JET, JT-60SA, ITER and DEMO.

the DEMOnstration power plant (DEMO), which is in the conceptual design phase, are claimed to have Q-value of 10 and 25, respectively [4, 5]. Figure 1.1¹ shows a comparison of the plasma volumes (in m³) in JET, JT-60SA, ITER, and DEMO.

In parallel to these developments, the mega ampere spherical tokamak (MAST), operational from 2000 to 2013, was designed to explore the benefits of the spherical tokamak design, including better plasma stability and confinement at higher pressures. The upgraded version, MAST-U, began operation in 2020 and aims to further investigate the spherical tokamak concept with advanced features and higher performance. The research and advancements from MAST and MAST-U are critical stepping stones towards the Spherical Tokamak for Energy Production (STEP) project, which aims to deliver a prototype fusion energy plant by the 2040s [6].

1.2 HIVE experimental facility

The Heat by Induction to Verify Extremes (HIVE) is an experimental facility at the UKAEA's Culham site designed to test plasma-facing components (PFCs) under the high thermal loads they will encounter in a fusion reactor. The PFCs are located in close proximity to the plasma and must be designed to endure these extreme conditions over the long term. The HIVE facility conducts multi-physics experiments in a vacuum chamber, that involve induction heating to simulate the thermal loads experienced by PFCs during normal fusion device operation, while using pressurised coolant to replicate the primary cooling system, which is responsible for removing heat from the reactor components.

¹© 2024 Fusion for Energy (F4E). All rights reserved. Used with permission from Fusion for Energy (F4E).

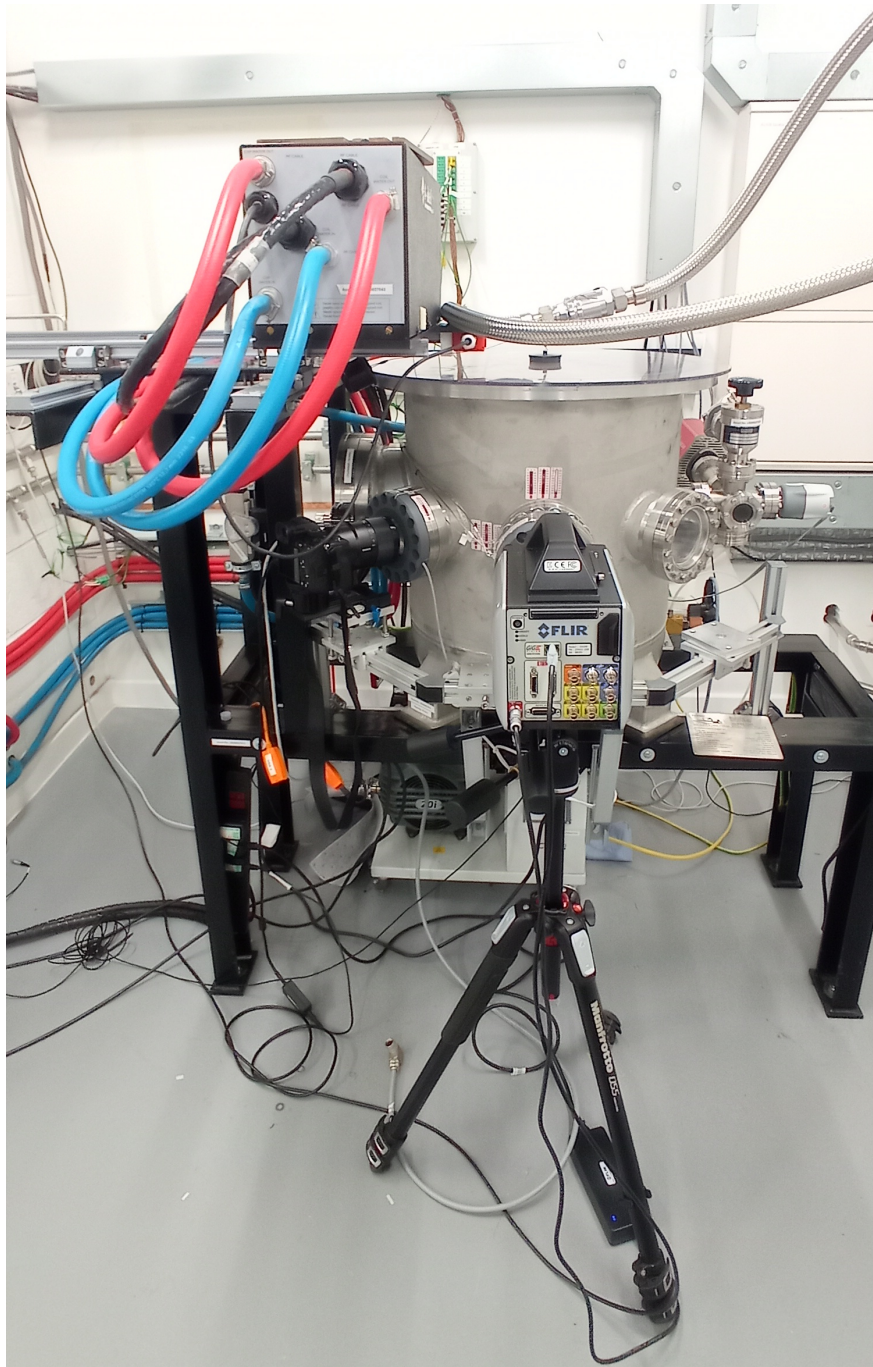


Figure 1.2: Photograph of the HIVE experimental setup taken in the lab.

Figure 1.2 shows the experimental setup including the cylindrical vacuum vessel with six ports around its circumference and an infrared camera used to record the experiment. Two ports are assigned to vacuum pumping and monitoring, while the remaining four provide a range of viewing angles.

HIVE can replicate a heat flux of up to 15 MW/m^2 , which is significant given that the

conceptual reactors like DEMO aim for a peak heat flux of up to 25 MW/m^2 [7]. This means that HIVE can simulate the extreme thermal conditions expected in future fusion reactors, allowing for accurate testing and validation of PFCs.

While there are various techniques for collecting experimental data in HIVE, such as infrared (IR) cameras, the primary method is through thermocouples. These probes are attached to the surface of a component to record pointwise temperature measurements. However, the number of thermocouples available for use in HIVE is limited, and their placement is constrained to ensure they do not interfere with the induction coil [8].

1.3 CHIMERA experimental facility

Fusion reactor components must undergo rigorous testing in a representative environment before deployment. Certain load conditions and operational scenarios for PFCs can only be accurately tested within a reactor environment. There are several challenges associated with prototype fusion reactors such as harsh environment and challenging access, thus the data collection will be limited and potentially unreliable. Thus design validation and predictive maintenance of sample under test (SUT) must be carried out in small scale test facilities.

The Combined Heating and Magnetic Research Apparatus (CHIMERA), is a unique SUT test facility, currently under design and construction at the UKAEA site in South Yorkshire. Unlike HIVE, CHIMERA is designed to test component prototypes under conditions that closely mimic those of a fusion reactor. CHIMERA will test SUT using a combination of heat, representative magnetic fields that recreate the overall magnetic environment of a tokamak, and water cooling, similar to HIVE. There are also plans to upgrade to liquid metal cooling in the future. In contrast, HIVE focuses on replicating high heat flux conditions using induction heating. CHIMERA will play an importance role in qualification of PFCs. Once the power plant is functional, simulations and surrogate models, continually updated with the historical data, will be used to provide live monitoring and predictive maintenance [9, 10].

1.4 Testing of fusion components

Both the HIVE and CHIMERA facilities aim to gain additional knowledge about fusion engineering components, such as the SUT, including informing design choices and validating computational simulations under extreme conditions that closely represent the reactor environment. Usually, in an experimental campaign, these SUT are tested under different conditions specified by the

designer. These conditions can include prescribed values of a field variable, such as temperature or thermal gradient applied to a certain part of the SUT, while ensuring these field variables stay below the service limit of the SUT. These tests help us study and understand the behaviour of several quantities related to the SUT. These studies are often referred to as the design of physical experiments (DoPE).

Modelling problems where the target value of field variables is known but the experimental parameters that cause the outcome are not known or are elusive can be incredibly challenging. These problems are conventionally referred to as inverse problems. Inverse problems involve determining the unknown parameters that produce a known set of outputs or effects. It is impractical to determine experimental parameters based solely on experience, intuition, and trial and error. This approach is not only highly inaccurate and time-consuming but also computationally expensive, particularly when dealing with a large number of experimental parameters.

The knowledge of thermo-mechanical behaviour of a PFC plays a critical role in the testing. Unfortunately, due to operational constraints, the inclusion of diagnostic tools are very limited. As a result, the data collection is limited and noisy. The development of techniques to infer unknown boundary conditions (BCs) or material properties from the limited or sparse data of a field variable would be useful in design choices and validation of computational simulations.

1.5 Inverse problem and digital twin

An inverse problem involves determining unknown causes or system parameters from observed effects or outcomes. This contrasts with a forward problem, where outcomes are predicted based on known parameters. Inverse problems are often ill-posed, meaning they may not satisfy one or more of the conditions of well-posed problems as defined by Hadamard [11]:

- **Existence:** Inverse problems may lack a solution due to insufficient or incomplete data, requiring additional information or approximations.
- **Uniqueness:** Non-uniqueness means that multiple solutions can exist, necessitating additional constraints to identify the correct solution which is physically relevant.
- **Stability:** Solution may not depend continuously on the input data, meaning small changes in data can lead to large changes in the solution.

These three properties make inverse modelling an incredibly difficult challenge. In this thesis, we focus on an inverse heat transfer problem (IHTP), a well known inverse problem in science and

engineering, where the unknown BCs and material properties are inferred from sparse measurements. A potential extension of IHTP can be the inference of the full temperature field from sparse measurements.

This extension can be used as a digital twin. A digital twin is a virtual representation of a physical system that is continuously updated with real-time data and simulations to replicate the system's behaviour accurately. To create a digital twin in the context of thermal problems, we can integrate real-time data from sensors to infer the full temperature field and perform rapid solution reconstruction. Specifically, after inferring the temperature field from sparse measurements at specific intervals during the experiment, we can quickly reconstruct the temperature field. This will enable us to monitor if the temperature exceeds the maximum limit and determine if the material properties reach critical values over time. By leveraging real-time sparse measurements, the digital twin can facilitate real-time monitoring, predictive insights, and support decision making to optimise the experimental campaigns and ensure safety.

1.6 Aims and Objectives

The primary aim of this thesis is to investigate the suitability of PINNs for solving inverse problems and to extend their application to construct a fundamental physics-informed digital twin platform for replicating physical experiments. PINNs are a class of ML models that embed physical laws, represented by PDEs, into the neural network's loss function. This approach allows them to solve forward and inverse problems efficiently, even with limited or sparse data.

We specifically chose PINNs because they do not require large training datasets for extensive pre-training. This characteristic makes PINNs particularly suitable for scenarios where data collection is challenging and generating simulation data is not feasible due to unknown or elusive BCs. PINNs have demonstrated their simplicity in solving inverse problems without using complicated algorithms that require problem-specific tuning. However, PINNs are relatively new and may have quirks that are not yet fully understood, necessitating a comprehensive study of their capabilities and limitations. Since we are performing solution reconstruction without extensive pre-training, we must incorporate additional information to obtain a unique and physically relevant solution. To achieve these aims, the following objectives are identified:

1. Conduct a comprehensive review of PINNs, identifying their limitations and potential improvements.
2. Solve challenging thermal problems, including those with discontinuities in the solution or input parameters, as well as parametric problems.

3. Determine the optimal set of hyperparameters for the aforementioned problems through hyperparameter tuning.
4. Develop a PINN-based model or workflow for inverse modelling without extensive pre-training.
5. Construct a fundamental physics-informed digital twin to replicate physical experiments from sparse measurements leveraging the inverse model developed in objective 4.

1.7 Outline of the thesis

This thesis comprises of 7 chapters as follows:

- **Chapter 1: Introduction** presents a brief introduction of fusion energy and history of major fusion devices. A brief discussion of HIVE and CHIMERA experimental facility is also provided. The role of inverse modelling and digital twinning has been discussed in the context of testing the fusion component prototypes. The current limitations associated with the testing of fusion components are used as motivation for the thesis. Lastly, the aim and objectives of this thesis are stated setting up a stage for the rest of the thesis.
- **Chapter 2: Physics-informed Neural Networks** presents the fundamental aspects of NNs and PINNs in detail. A detailed study of issues associated with the baseline PINNs and the potential remedies are also presented for solving forward problems. Several test cases showcasing applications of PINNs to forward problems such as the Burgers equation, Allen-Cahn equation and the Lid driven cavity are also presented. The chapter primarily focuses on solution to parametric problems, approaches to tackle stiff-PDEs and problems involving complex geometries.
- **Chapter 3: Solving stiff-PDEs using PINNs** focuses on problems with conflicting BCs at adjacent edges and corners. Investigation was conducted on four test cases that included heat conduction problems with parametric conductivity and parametric geometry with discontinuous BCs. The test cases were solved using a number of PINN frameworks, discussed and analysed the results against the FEM solution. A discussion of the current challenges associated with PINNs is also presented.
- **Chapter 4: Hyperparameter tuning of PINNs** conducts a hyperparameter tuning of PINNs as the accuracy of the NN is contingent upon the selection of appropriate hyperparameters. Implementing PINNs for problems with discontinuous BCs or discontinuous

PDE coefficients is a challenge and thus can be thought of as the continuation of chapter 2. A manual tuning with more than 100 parameter settings was performed to find the optimal neural network architecture, number of hidden layers, learning rate and activation function for heat conduction problems with a discontinuous solution leveraged from chapter 2.

- **Chapter 5: Inverse solution reconstruction** presents a PINN-based workflow to reconstruct field variables such as velocity, temperature, or pressure from sparse measurements. The workflow effectively reduces the need for both extensive pre-training and explicit knowledge of BCs, as well as the deployment of an impractical number of sensors in experimental settings. The workflow is rigorously evaluated against two steady-state heat conduction problems with different unique boundary conditions, and a problem governed by the Helmholtz equation. This workflow simplifies the process of solving inverse problems by minimising the reliance on pre-existing experimental datasets. It enhances the feasibility of conducting experiments in areas where data acquisition is a challenge.
- **Chapter 6: HIVE digital twin** showcases foundational development of a physics-informed digital twin for the HIVE sample using the PINN-based workflow discussed in chapter 5. A digital twin was developed to reconstruct full steady-state temperature field of the HIVE sample after each increment in the input current during the experiment. This enable us to determine if the temperature exceeded the maximum limit and the material properties reach critical values over time.
- **Chapter 7: Conclusions and future work** presents an overview of the work and the main achievements of the thesis, highlighting its benefits to the wider scientific community. The chapter concludes the thesis by summarising the research findings and their implications. It also discusses potential future work that could further enhance the methodologies and frameworks used, particularly in obtaining results with the HIVE digital twin.

1.8 Dissemination of research

The research conducted during this thesis has been disseminated through a combination of peer-reviewed publications, works under review, and conference presentations. While not all works are published or peer-reviewed at this stage, they have been systematically incorporated into the relevant chapters of this thesis to provide a cohesive flow of the research progression.

1.8.1 Research papers

1. Sharma, P., Evans, L., Tindall, M., & Nithiarasu, P. (2023). Stiff-PDEs and Physics-Informed Neural Networks. *Archives of Computational Methods in Engineering*, 1-30. doi: 10.1007/s11831-023-09890-4.
2. Sharma, P., Evans, L., Tindall, M., & Nithiarasu, P. (2023). Hyperparameter selection for physics-informed neural networks (PINNs)—Application to discontinuous heat conduction problems. *Numerical Heat Transfer, Part B: Fundamentals*, 1-15. doi: 10.1080/10407790.2023.2264489
3. Sharma, P., Tindall, M., & Nithiarasu, P. (2024). Physics-informed neural networks for solving partial differential equations. In *Artificial Intelligence in Heat Transfer: Advances in Numerical Heat Transfer (Vol. 6)*. CRC Press, Taylor and Francis Group LLC.
4. Sharma, P., Tindall, M., & Nithiarasu, P. (n.d.). Sparse measurements to solution reconstruction using Physics-informed neural networks (PINNs). Under review.

1.8.2 Conference presentations

1. PINNs for solving forward and inverse thermal problems. AI Conference, Centre for Doctoral Training in Artificial Intelligence, Machine Learning and Advanced Computing (AIMLAC), Swansea University. 7-9 Jun 2023.
2. Hyperparameter Tuning of PINNs. UK Association for Computational Mechanics Conference, University of Warwick. 19-21 Apr 2023
3. Solution reconstruction of HIVE sample using physics-informed neural networks. ThermaE-Comp, Montenegro. 9-11 Sep 2024

Chapter 2

Physics-informed Neural Networks

2.1 Introduction

Conventional numerical techniques for solving PDEs have been a cornerstone in engineering design for years. However, their implementation poses challenges for a variety of problems. Some of the common issues encountered include: mesh dependency of solutions, computational expense in high-dimensional parametric solutions, stress concentration at sharp corners, challenges in achieving convergence and stability, and difficulties in generating adaptive meshes.

Conventional numerical techniques such as finite elements and finite volumes were primarily developed for forward problems. They encounter significant challenges when solving inverse problems. The ill-conditioned nature of inverse problems, need to integrate noisy experimental data and the existence of non-unique solutions makes these problems intractable for conventional numerical techniques. This chapter does not cover these challenges associated with inverse problems, instead focuses on the advancements and applications of PINNs in forward problems.

Although early efforts were made to solve differential equations with neural networks, these were constrained by the limitations of smaller-scale neural networks (NNs) and less efficient optimisers [12–15]. The recent advancement in existing algorithms and computing power have led to significant achievements in the field [16–18]. Rudd *et al.* [16] introduced the constrained integration approach, which integrates PDE constraints into ANNs for solving complex equations efficiently. Raissi *et al.* [17] extended this with numerical Gaussian processes, combining Bayesian inference with numerical methods to solve time-dependent and nonlinear PDEs while providing uncertainty quantification. Additionally, Raissi *et al.* [18] demonstrated the use of deep learning for fluid mechanics, accurately modelling vortex-induced vibrations and showcasing the potential of machine learning for non-linear problems.

The numerical solution of ordinary differential equations (ODEs) and PDEs using NNs [12–19] has been a key area of interest for researchers. This is because the NNs behave like meshless solvers and can be scaled to higher dimensions without the need of additional algorithms.

Recently, Raissi *et al.* [20] introduced the groundbreaking concept of PINNs, a method to seamlessly integrate both data and PDE within a deep learning framework. They demonstrated its capability to effectively solve both forward [21–24] and inverse problems [25–27].

Although, NNs have been present in literature for a long time, they started gaining immense popularity only in the last decade due to the recent advancement in automatic differentiation and the availability of open-source ML libraries such as PyTorch [28] and TensorFlow [29]. Among these advancements, automatic differentiation [30] plays a pivotal role by producing highly accurate derivatives, which enhances the accuracy of PINNs by enabling the calculation of the residual of the PDE.

PINNs mark a significant evolution in Scientific machine learning (SciML) community, by embedding the physical laws described by the differential equations in the NN. Traditional numerical methods, while foundational in engineering, grapple with challenges such as mesh dependency and computational burdens in high dimensions. Unlike conventional numerical techniques, PINNs are inherently meshless, effectively addressing several aforementioned issues such as mesh dependency and adaptive mesh generation.

Additionally, while conventional methods like FEM and FDM can reuse meshes or leverage solutions from similar problems to accelerate computations, PINNs offer a unique advantage by directly embedding physical laws into the learning process. This allows them to utilise transfer learning more effectively, accelerating the solution of new problems by leveraging information from previously solved problems. This represents a significant advantage over traditional NN-based PDE solvers, which typically requires a large number of ground truth, such as simulation results, for model training.

2.2 Physics-driven vs. Data-driven approaches

There is a plethora of methodologies for tackling physics-based problems, leveraging both conventional numerical solvers and machine learning (ML) models. Before the advent of PINNs, the predominant approaches were either physics-driven or data-driven. As depicted in Figure 2.1, physics-driven techniques such as conventional numerical solvers, are on the left, whereas purely data-driven methods, including traditional supervised ML models like NNs and regression algorithms that rely on large labelled datasets, are on the right. These methods often focus on

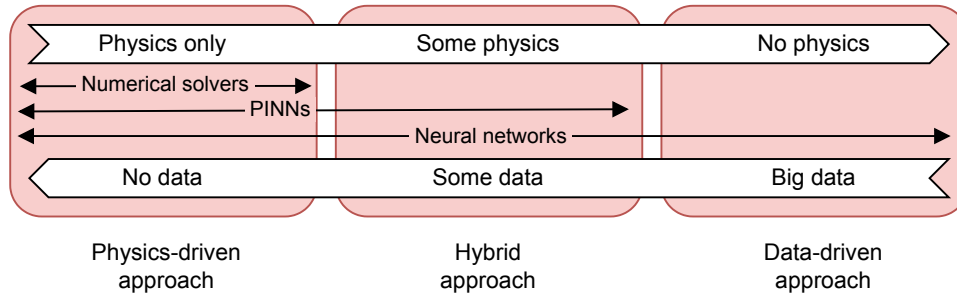


Figure 2.1: A broad classification of different approaches for solving physics problems.

pattern recognition and prediction, with minimal or no incorporation of physical laws governing the problem.

There are two distinct data-driven methods for enforcing the underlying physics: physics-guided NN (PGNN) and physics-encoded NN (PENN). PGNNs follow a classical supervised learning framework, constructing surrogate models from data derived from experiments and simulations. These models typically necessitate extensive datasets to achieve generalisation [31, 32].

PENN-based models aim to directly encode underlying physical principles within the NN's architecture, offering an advantage in scenarios where traditional methods struggle with complex boundary conditions, unknown source terms, or noisy data. Among these, two notable approaches stand out: physics-encoded recurrent convolutional NN (PERCNN) [33] and neural ordinary differential equations (NeuralODE) [34].

The PERCNN integrates the non-linear systems directly into the NN by replacing the traditional activation function with a novel element-wise product operation. This modification allows the NN to directly encode the dynamics of physical systems into its computational process, enabling it to handle complex, non-linear behaviours more effectively than traditional NNs.

In contrast, NeuralODE reimagines the structure of NNs to parallel the behaviour of ordinary differential equations (ODEs) that can be solved using Euler's method. By designing the NN's layers to represent discrete steps in solving an ODE, NeuralODE allows for the direct application of numerical methods within the NN. This architecture creates a bridge between numerical analysis and machine learning.

PINNs belong to the region between physics-driven and hybrid approach. This signifies that PINNs are capable of solving PDEs without needing additional ground truth, but they can seamlessly integrate noisy experimental data which is a significant advantage over conventional numerical solvers.

Table 2.1: Conventional numerical techniques vs PINNs

	Conventional numerical techniques	PINNs
Basis function	Piecewise polynomial	Neural network
Solution methodology	Numerical approximation & iterative methods	Optimisation problem
PDE embedding	Discretised equations	Loss function
Geometric representation	Mesh	Point cloud

2.3 How PINNs are different from conventional numerical techniques?

Conventional numerical techniques such as finite elements (FEM) and finite volumes (FVM) were primarily developed for forward problems [35–37]. In these methods, the domain is discretised into a mesh consisting of elements (in FEM) or control volumes (in FVM), with corners or boundaries defined by nodes. In FEM, each element uses a basis function, often described by a piecewise polynomial function, to interpolate the solution within the element based on the nodal values of the field variable. In contrast, FVM divides the domain into control volumes, and the solution is directly approximated within these volumes.

The PINNs are NNs that model the forward problem as an optimisation problem. Instead of discretising the PDE, PINNs formulate them as a part of the loss function. Traditional mesh-based discretisation is replaced with a point cloud throughout the domain. The solution is then inferred at these points once the network has been fitted to minimise the loss function.

PINNs are fundamentally different from conventional PDE solvers like FDM or FEM in how they represent the system of equations. In PINNs, the number of equations (or data points) exceeds the number of unknowns (or network parameters), transforming the problem into a regression task rather than a direct solution of the PDEs. On the other hand, the conventional PDE solvers, such as FEM, often solves the system of equations with the number of equations and unknowns being equal. The key distinctions between conventional numerical techniques and the PINNs are summarised in Table 2.1.

2.4 Neural networks for regression

NNs are nonlinear and non-convex regression frameworks with exceptional predictive capability, widely known as universal approximators of continuous functions [38, 39]. They are known for

their ability to learn and generalise very complicated information [40]. This section discusses the evolution of NNs.

One of the simplest regression models is the linear regression. In the case of linear regression, the hypothesis or the trial function is the linear combination of weights and features. Let us assume a dataset with n samples consisting of m features with a bias term $\mathbf{X} = \{1, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_m\}$ and one label \mathbf{y} . The linear regression hypothesis for the dataset can be written as follows

$$\hat{\mathbf{y}} = w_0 + w_1\mathbf{x}_1 + w_2\mathbf{x}_2 + \dots + w_m\mathbf{x}_m = \mathbf{W}^T \mathbf{X} \quad (2.1)$$

where, the trainable weights $\mathbf{W} = \{w_0, w_1, w_2, \dots, w_m\}$ are tweaked such that the mean squared error or any loss metric of the ground truth \mathbf{y} against the predicted output $\hat{\mathbf{y}}$ is minimised.

In principle, we can introduce a hypothesis with a linear combination of weights with nonlinear features too. The main drawback of linear regression is underfitting because it essentially fits a linear equation onto the data points [41, 42].

A simple technique to prevent underfitting is to pass the output of linear regression hypothesis (Equation 2.1) to specifically chosen nonlinear function, often referred to as the activation function (σ) [43]. These activation functions, for example, sigmoid, hyperbolic tangent, softmax are continuous and sometimes are infinitely differentiable in the domain of real numbers. The resulting hypothesis (Equation 2.2) is referred to as an artificial neuron (Figure 2.2) [44]. It transforms the linear regression hypothesis into a nonlinear feature space.

$$\hat{\mathbf{y}} = \sigma(w_0 + w_1\mathbf{x}_1 + w_2\mathbf{x}_2 + \dots) = \sigma(\mathbf{W}^T \mathbf{X}) \quad (2.2)$$

The artificial neuron is the building block of a NN. DNNs have multiple hidden-layers, each hidden-layer consisting of multiple artificial neurons thus increasing the predictive capability even further. Each hidden layer transforms the feature space into a more complex feature space [45]. It is an open question in interpretable machine learning to explain mathematically, how these nonlinear transformations influence the output [46]. Figure 2.3 shows the schematic diagram of a fully connected neural network (FCNN) with 2 hidden layers.

In Figure 2.3, there are 4 layers, one input layer, two hidden layers and one output layer. Both hidden layers consist of three artificial neurons, also known as the activated neurons or simply the nodes. In a FCNN, each node has its own set of weights or parameters.

Let us define some notation, $a_i^{(k)}$ is the i th activated neuron in the k th layer. w^k is the weight matrix controlling the nonlinear mapping between k th layer and $(k+1)$ th layer. If the network has s_k activated neurons in k th layer and $s_{(k+1)}$ activated neurons in $(k+1)$ th layer then w^k will

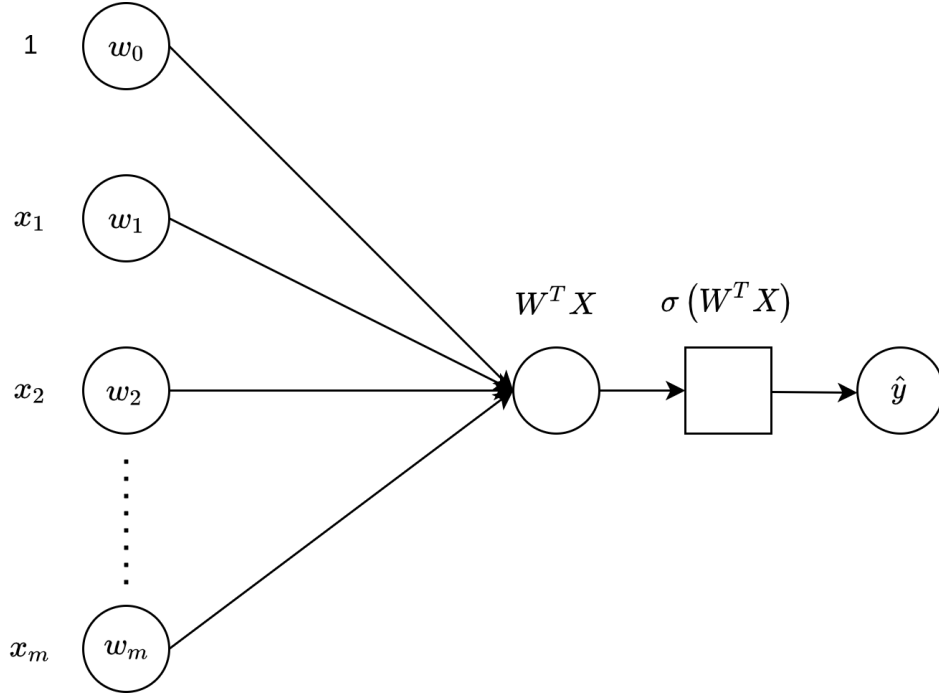


Figure 2.2: Schematic diagram of an artificial neuron. Here, $\mathbf{X} = \{1, x_1, x_2, \dots, x_m\}$ is the input dataset, $\mathbf{W} = \{w_0, w_1, w_2, \dots, w_m\}$ is the vector of trainable weights/ parameters, σ is the activation function and \hat{y} is the predicted output.

be of dimension $s_{(k+1)} \times s_k$.

2.4.1 Forward pass

The first step in training of a NN is to forward pass the inputs \mathbf{X} to obtain the predicted output \hat{y} . Let $a_i^{(k)} = \sigma(z_i^{(k)})$ such that z is the term without the activation. Now, the forward pass for the FCNN in Figure 2.3 can be written as follows:

$$\begin{aligned}
 \mathbf{z}^{(2)} &= \mathbf{w}^{(1)} \mathbf{X} \\
 \mathbf{a}^{(2)} &= \sigma(\mathbf{z}^{(2)}) \\
 \mathbf{z}^{(3)} &= \mathbf{w}^{(2)} \mathbf{a}^{(2)} \\
 \mathbf{a}^{(3)} &= \sigma(\mathbf{z}^{(3)}) \\
 \hat{\mathbf{y}} &= \mathbf{w}^{(3)} \mathbf{a}^{(3)}.
 \end{aligned} \tag{2.3}$$

In the context of NN used for regression problems, the last layer typically does not have an activation function. This is because, in regression tasks, we aim to predict continuous values

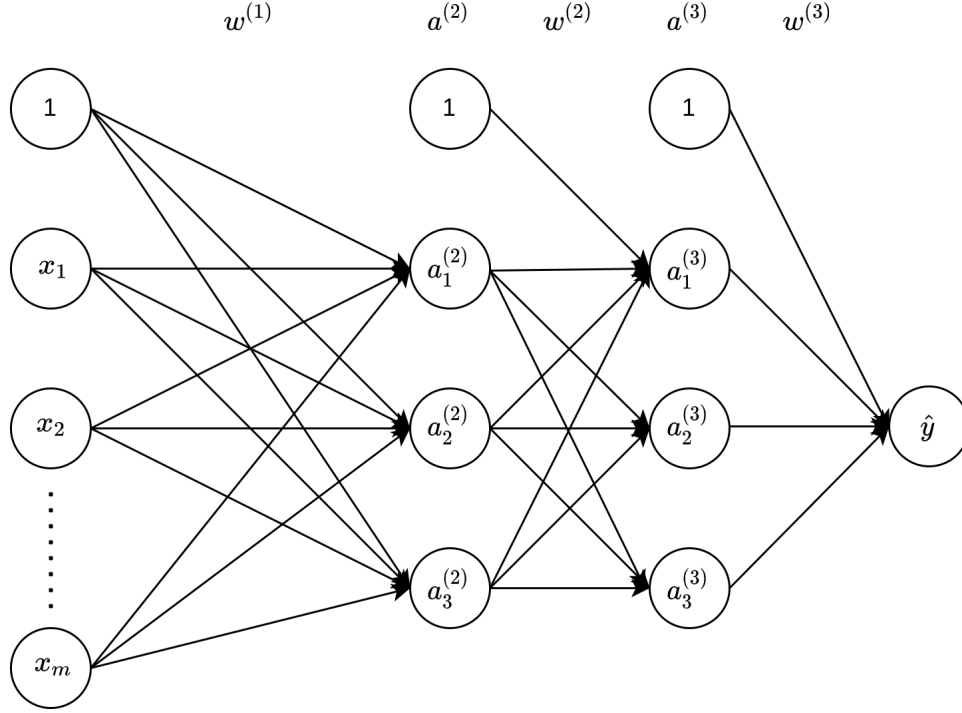


Figure 2.3: Schematic diagram of a fully connected neural network (FCNN) with 2 hidden layers. Here, $\mathbf{X} = \{1, x_1, x_2, \dots, x_m\}$ is the input dataset, $\{w^{(1)}, w^{(2)}, w^{(3)}\}$ is the vector of trainable weights/ parameters for each layer, $\{a^{(2)}, a^{(3)}\}$ is the vector of activated layers and \hat{y} is the predicted output.

that can span a wide range. By not applying an activation function to the last layer, we ensure that the output values are not restricted to a specific range, allowing the network to produce any real number as a prediction, which is essential for accurately modelling continuous data. If we combine the steps in Equation 2.3, the hypothesis of a FCNN with 2 hidden layers can be written as follows:

$$\hat{y} = w^{(3)} \sigma(w^{(2)} \sigma(w^{(1)} \mathbf{X})). \quad (2.4)$$

2.4.2 Backpropagation and automatic gradient

Once we have obtained the predicted output \hat{y} from the forward pass, we can calculate the difference from the ground truth y using some loss metric [47]. In general, for a regression problem, the mean squared error (Equation 2.5) is used as a loss metric.

$$J = \frac{1}{2n} \sum_{i=1}^n |y_i - \hat{y}_i|^2 \quad (2.5)$$

where n represents the number of samples. Now, the loss J is sent to the optimisers. Optimisers are algorithms that minimise the loss by updating the weights or parameters of the NN. There are many optimisers that are available in open-source libraries such as Pytorch and Tensorflow [48]. On the basis of order of the derivatives of the loss with respect to the weights, the optimisers can be divided mainly into two categories: first order and second order optimisers [49].

The first-order optimisers utilise the gradient of the loss with respect to weights of the NN. Similarly, second-order optimisers utilise the gradient as well as the Hessian of the loss with respect to the weights of the NN. Gradient-descent [50] and adaptive moment estimation (Adam) [51] are the most popular first-order optimisers. Whereas, limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) [52] is the only second-order optimiser that is still actively used in ML. L-BFGS is preferred over other second-order methods because it approximates the Hessian with limited memory, making it computationally efficient for high-dimensional ML problems. This efficiency allows it to leverage second-order information for faster convergence without the high memory cost of methods like Newton’s method.

Now that the optimisers require derivatives, we need to compute the derivatives efficiently and accurately. There are three main techniques that have been successfully used in machine learning: numerical [53], symbolic [54] and automatic differentiations [30]. After the release of Tensorflow 1.0 in 2015, static computational graphs was the standard data structure for representing NNs, which was later substituted with dynamic computational graphs in PyTorch [55, 56]. These libraries use forward-mode or reverse-mode automatic differentiation to compute the derivatives within a computational graph [57]. Although automatic differentiation can be used to calculate any derivative, in most machine learning applications, it computes the derivative of the loss with respect to weights using the chain rule [58] in differential calculus. The derivatives of basic functions, such as x^2 and e^x , are hard-coded in the system. Using these pre-defined rules, the system dynamically calculates derivatives through the computational graph [59].

Let us say that we are using the gradient-descent method (Equation 2.6) for optimising the weight/parameters.

$$\mathbf{w}^{(k)} := \mathbf{w}^{(k)} - \alpha \frac{\partial J}{\partial \mathbf{w}^{(k)}} \quad (2.6)$$

where k is the k th layer and α is a constant often referred to as the learning rate, a hyperparameter. It tells the optimiser the amount of perturbation the weights should attain in each epoch [60]. An epoch is a complete cycle of forward pass and backpropagation on the whole dataset.

The gradient of the loss J with respect to the weights in each layer can be computed using the chain rule as follows:

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{w}^{(3)}} &= \frac{\partial J}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{w}^{(3)}} \\
\frac{\partial J}{\partial \mathbf{w}^{(2)}} &= \frac{\partial J}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{a}^{(3)}} \frac{\partial \mathbf{a}^{(3)}}{\partial \mathbf{z}^{(3)}} \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{w}^{(2)}} \\
\frac{\partial J}{\partial \mathbf{w}^{(1)}} &= \frac{\partial J}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{a}^{(3)}} \frac{\partial \mathbf{a}^{(3)}}{\partial \mathbf{z}^{(3)}} \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{a}^{(2)}} \frac{\partial \mathbf{a}^{(2)}}{\partial \mathbf{z}^{(2)}} \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{w}^{(1)}}
\end{aligned} \tag{2.7}$$

As mentioned earlier, the analytical expressions of basic functions are hardcoded in machine learning libraries. The derivative of the loss with respect to the weights is computed by decomposing it into these hard-coded basic functions using the chain rule. Once, these derivatives are calculated we plug them into Equation 2.6 to update the weights/parameters. The process is repeated for a number of epochs until the loss J goes below some predefined tolerance.

2.4.3 Complexity of analytical derivatives in neural networks

The complexity of these derivatives increases with the size of the NN. To exemplify this complexity, consider the derivative of the output of a NN with respect to its input. Using a simple neural network model with one hidden layer and a sigmoid activation function, we can analytically derive the expression for the derivative. For simplicity we restrict ourselves to a single input x and a single neuron in each layer. Let x be the input, w_1 and w_2 be the weights of the first and second layers respectively, and b_1 and b_2 be the biases. The network output \hat{y} can be expressed as:

$$\hat{y} = \sigma(w_2 \cdot \sigma(w_1 \cdot x + b_1) + b_2).$$

where σ denotes the sigmoid function. The derivative of \hat{y} with respect to x is:

$$\frac{\partial \hat{y}}{\partial x} = \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial x}$$

where $z_2 = w_2 \cdot a_1 + b_2$ and $a_1 = \sigma(w_1 \cdot x + b_1)$. Expanding this, we have:

$$\frac{\partial \hat{y}}{\partial x} = \frac{w_1 w_2 e^{-b_1 - w_1 x} e^{-b_2 - \frac{w_2}{e^{-b_1 - w_1 x} + 1}}}{(e^{-b_1 - w_1 x} + 1)^2 \left(e^{-b_2 - \frac{w_2}{e^{-b_1 - w_1 x} + 1}} + 1 \right)^2}.$$

Let's examine how the complexity increases as the number of hidden layers is increased. By maintaining a single input, a single neuron in each hidden layer, and a single output, but increasing the number of hidden layers to two, we obtain a significantly more complicated derivative. The

derivative of \hat{y} with respect to x for a NN with two hidden layers is:

$$\frac{\partial \hat{y}}{\partial x} = \frac{w_1 w_2 w_3 e^{-b_1 - w_1 x} e^{-b_2 - \frac{w_2}{e^{-b_1 - w_1 x} + 1}} e^{-b_3 - \frac{w_3}{e^{-b_2 - \frac{w_2}{e^{-b_1 - w_1 x} + 1} + 1}}}}{(e^{-b_1 - w_1 x} + 1)^2 \left(e^{-b_2 - \frac{w_2}{e^{-b_1 - w_1 x} + 1}} + 1 \right)^2 \left(e^{-b_3 - \frac{w_3}{e^{-b_2 - \frac{w_2}{e^{-b_1 - w_1 x} + 1} + 1}} + 1 \right)^2}.$$

These intricate derivatives showcase the computational cost involved in training NNs, underscoring the importance of automatic differentiation in ML libraries.

2.5 The baseline PINN

The baseline PINN takes in the independent variables as the input and gives out the dependent variables as the output. The construction of the loss function depends on the nature of the problem, such as whether it is time-dependent or requires specific BCs. The baseline PINN accurately predicted the solution of forward problems such as the 1D Burgers equation, 1D Schrodinger equation, 1D Allen-Cahn equation and inverse problems such as recovering the 2D Navier-Stokes equation from the finite element generated flow past a cylinder simulation.

Consider a well-posed PDE problem as follows:

$$\begin{aligned} \mathbf{u}_t + \mathcal{N}_x[\mathbf{u}] &= 0, \quad \mathbf{x} \in \Omega, t \in [0, T] \\ \mathbf{u}(\mathbf{x}, 0) &= h(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega \\ \mathbf{u}(\mathbf{x}, t) &= g(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega, t \in [0, T] \end{aligned} \tag{2.8}$$

where the first equation represents the PDE with a temporal derivative \mathbf{u}_t and a spatial derivative operator $\mathcal{N}_x[\mathbf{u}]$. Here, $\mathbf{u}(\mathbf{x}, t)$ is the dependent variable, where \mathbf{x} and t denote the independent spatial and temporal variables, respectively. The Ω and $\partial\Omega$ denote the spatial domain and the boundary of the problem. The function $g(\mathbf{x})$ specifies the BCs and $h(\mathbf{x}, 0)$ denotes the initial condition (IC) at $t = 0$.

The inputs to a PINN for a 2D time dependent problem, are the spatio-temporal coordinates denoted by x, y, z and t . Unlike the mesh generation in conventional numerical methods like FEM, where the structure of the grid can significantly influence solution, the sampling of coordinates for a PINN can be conducted in a more arbitrary manner. There are certain techniques to effectively reduce the number of random points needed while still ensuring comprehensive domain coverage as discussed in Section 2.6.1. These random points are then fed into the NN, as illustrated in

Figure 2.4. The baseline PINN employs a feed-forward NN (FNN), which comprises several fully connected layers leading to the predicted output, represented by \hat{u} [61].

2.5.1 Discrete loss function

Similar to traditional supervised ML techniques, in PINNs, the predicted output \hat{u} plays a crucial role in formulating the constraints, which are represented through a loss function [62]. Unlike simpler models, PINNs require multiple loss functions to simultaneously satisfy the PDE, the BCs and the IC, if the problem is transient. This approach results in a multitask loss function, comprising the individual loss terms (\mathcal{L}_{PDE} , \mathcal{L}_{BC} , \mathcal{L}_{IC}) and resulting in the total loss (\mathcal{L}) term, defined as follows:

$$\mathcal{L} = \lambda_{PDE}\mathcal{L}_{PDE} + \lambda_{BC}\mathcal{L}_{BC} + \lambda_{IC}\mathcal{L}_{IC} \quad (2.9)$$

$$\begin{aligned} \mathcal{L}_{PDE} &= \frac{1}{N_r} \sum_{i=1}^{N_r} |\hat{u}_t(\mathbf{x}_i, t_i) + \mathcal{N}_x[\hat{u}(\mathbf{x}_i, t_i)]|^2 \\ \mathcal{L}_{BC} &= \frac{1}{N_b} \sum_{i=1}^{N_b} |\hat{u}(\mathbf{x}_i, t_i) - g(\mathbf{x}_i, t_i)|^2 \\ \mathcal{L}_{IC} &= \frac{1}{N_0} \sum_{i=1}^{N_0} |\hat{u}(\mathbf{x}_i, 0) - h(\mathbf{x}_i)|^2 \end{aligned} \quad (2.10)$$

In these equations, N_r , N_b and N_0 represent the number of data points sampled to satisfy the PDE, BCs, and IC, respectively, as mentioned in Equation 2.9. The coefficients λ_{PDE} , λ_{BC} and λ_{IC} are weighting factors in Equation 2.9, that help in achieving better convergence and accuracy in the model. The PDE loss \mathcal{L}_{PDE} is the mean squared error (MSE) of the residual of the PDE, calculated with the help of automatic differentiation similar to Equation 2.4. Similarly, the BC loss \mathcal{L}_{BC} and the IC loss \mathcal{L}_{IC} are MSEs of the difference between $\hat{u}(x, y, z, t)$ and the known BC and IC at their respective locations. Figure 2.4 illustrates the overall architecture of a PINN for a 3D spatio-temporal problem.

In the baseline PINN framework, gradient-based optimisers are employed to minimise the total loss \mathcal{L} , by adjusting the weights of the NN during the training process [63]. Among these optimisers, Adam and L-BFGS are frequently utilised due to their efficiency in handling large-scale optimisation problems.

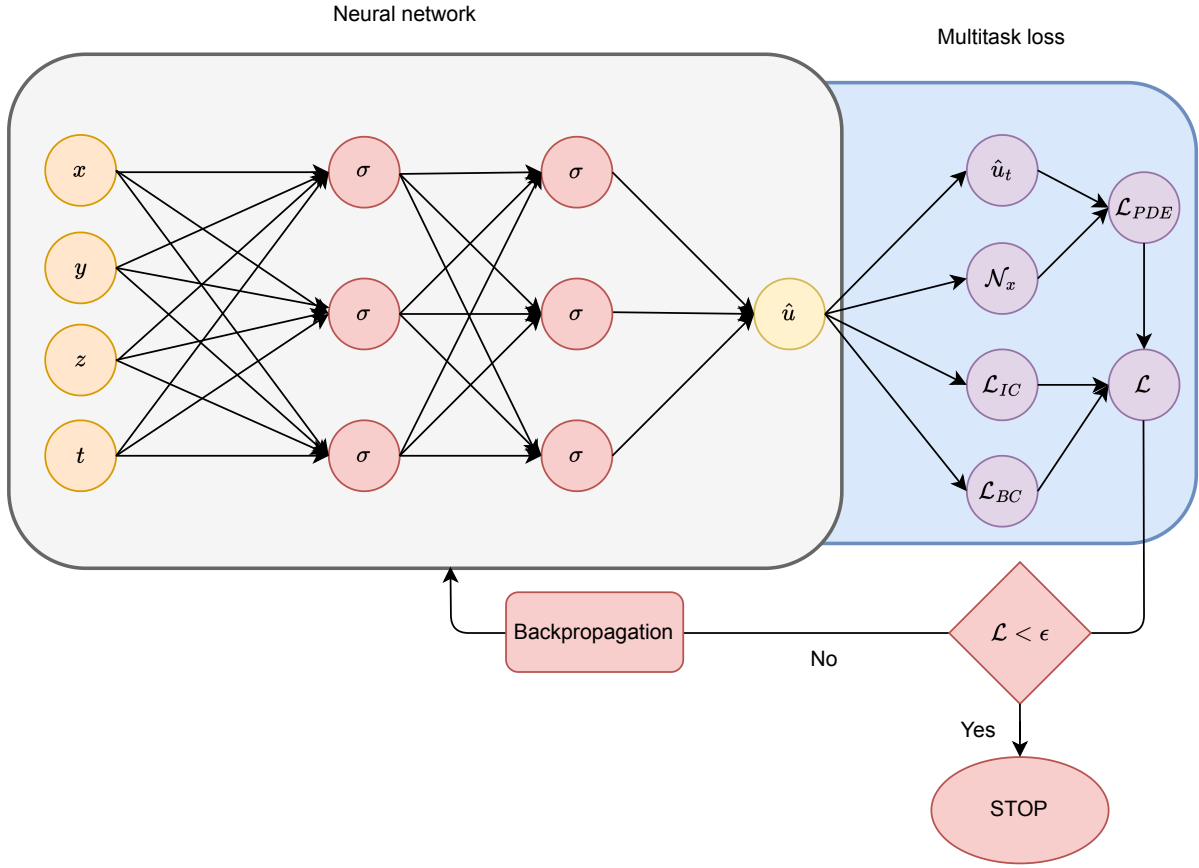


Figure 2.4: The architecture of a PINN with two hidden layers for a 3D spatio-temporal forward problem. Here the inputs are x, y, z and t , σ is the activated neuron and \hat{u} is the predicted output or the solution to the PDE.

Both Adam and L-BFGS bring distinct advantages to the training process of PINNs. Adam's adaptive learning mechanism can lead to faster convergence, especially in the early stages of training. Whereas, L-BFGS is often preferred in the later stages of training when fine-tuning around minima is required, as it can provide more accurate updates by approximating second-order curvature information.

Given that the problem is well-posed, there exists a unique solution [43]. The enforcement of the loss terms \mathcal{L}_{PDE} , \mathcal{L}_{BC} , \mathcal{L}_{IC} within the PINN framework contributes to maintaining the well-posedness of the problem, thereby facilitating the convergence towards a unique solution.

2.5.2 Reconstruction loss

A PINN framework, referred to as sparse-regulated PINNs, uses the experimental or the simulation data to provide ground truth at sparse locations in the spatio-temporal domain [64, 65]. The

addition of sparse ground truth helps the NN to accurately predict complex local variations in the solution. This introduces one more loss term called the reconstruction loss (Equation 2.11), because it points the optimiser towards the true solution.

$$\mathcal{L}_{reconstruct} = \frac{1}{N_d} \sum_{i=1}^{N_d} |\hat{u}(\mathbf{x}_i, t_i) - u_i^{true}|^2 \quad (2.11)$$

where N_d is the number of true solutions being provided. Bajaj *et al.* [66] showed that the reconstruction loss is not very helpful in preventing overfitting. Instead, NVIDIA Modulus uses the Adam optimiser with exponential decay of the learning rate α , so that as we go closer to the minima of the loss, the weights do not attain large perturbations [51].

2.5.3 Integral formulation of loss

NVIDIA Modulus, an open-source library uses a slightly different version of the discrete loss function [67, 68]. They define continuous or the integral losses for the BC and PDE loss as follows:

$$\begin{aligned} \mathcal{L}_{PDE} &= \int_{\Omega} (\hat{u}_t(\mathbf{x}_i, t_i) + \mathcal{N}_x[\hat{u}(\mathbf{x}_i, t_i)])^2 d\Omega \\ \mathcal{L}_{BC} &= \int_{\partial\Omega} (\hat{u}(\mathbf{x}_i, t_i) - g(\mathbf{x}_i, t_i))^2 d(\partial\Omega). \end{aligned} \quad (2.12)$$

Instead of approximating these integrals using deterministic numerical integration techniques, NVIDIA Modulus uses the Monte-Carlo integration, a non-deterministic integration technique [69, 70], resulting in equation 2.13. This helps in keeping a specific loss term proportional to its length/area/volume. For example, it doesn't allow a specific BC applied over a relatively larger area to dominate other BCs.

$$\begin{aligned} \mathcal{L}_{PDE} &= \frac{1}{N_r} \sum_{i=1}^{N_r} |\hat{u}_t(\mathbf{x}_i, t_i) + \mathcal{N}_x[\hat{u}(\mathbf{x}_i, t_i)]|^2 \int_{\Omega} d\Omega \\ \mathcal{L}_{BC} &= \frac{1}{N_b} \sum_{i=1}^{N_b} |\hat{u}(\mathbf{x}_i, t_i) - g(\mathbf{x}_i, t_i)|^2 \int_{\partial\Omega} d(\partial\Omega) \end{aligned} \quad (2.13)$$

2.5.4 Issues with the baseline PINN

The baseline PINN had several limitations such as scalability to higher dimensions, imbalanced magnitude of individual loss terms in the multiple task loss function, gradient explosion etc. A brief description of these issues has been given in this section and are explored further in the subsequent chapters.

The baseline PINN can be scaled to higher dimensions by modifying the architecture of the NN. A general approach is to deploy more activated neurons (~ 500) with very few hidden layers (~ 4). This approach helps in fitting complex local variations in the solution. Wang *et al.* [71] proposed a fully connected NN with two transformer networks [72–74], which projects the input variables to a high-dimensional feature space. Sirignano *et al.* [75] proposed a Deep Galerkin method (DGM) architecture influenced by the LSTM architecture [76]. Both these architectures attempt to encapsulate the complex local variations in the solution by using more complex NN architectures.

Spectral bias [77] is a learning bias of deep neural networks (DNNs) towards low-frequency functions, which causes convergence issues during training. Novel architectures such as Fourier networks [78], Modified Fourier networks [68, 79], Sinusoidal Representation networks (SiReNs) [80] were proposed to remove the spectral bias from computer vision problems.

Imbalanced loss terms appear when the magnitude of one loss term is significantly larger than other loss terms, resulting in early convergence of other loss terms in the multitask loss function. A common approach to address imbalanced loss terms is to multiply each loss term by a weighting parameter λ , which is chosen to adjust the contribution of each term, ensuring that no single term dominates the overall loss function [81]. Several frameworks such as self-adaptive PINNs [82] and the self-adaptive weight PINN [83]; algorithms such as learning rate annealing [71] and neural tangent kernel [84] were proposed to balance out the contribution of each term to the overall loss by introducing adaptive coefficients for each loss term.

Gradient explosion [85, 86] is a known issue while training NNs. The baseline PINN uses L-BFGS [52], a second order optimiser, which exhibits exploding gradients when it encounters sharp gradients [87].

2.5.5 Nature of solutions in PINNs

The PINNs were originally developed as solvers for PDEs. In conventional applications, once a PDE is solved within the specified domain with prescribed BCs and IC, further inference on new spatio-temporal locations is typically unnecessary. However, by incorporating validation

dataset, PINNs can be generalised to interpolate or extrapolate the field variables at new spatio-temporal locations. This predictive capacity aligns with the conventional ML techniques, where the availability of ground truth enables the model to learn and make accurate predictions on new spatio-temporal locations. It is important to note that this generalisation approach deviates from the traditional use-case of PDE solvers, which don't rely on ground truth data.

2.5.6 Loss metrics

In this thesis, we have utilised two primary loss metrics: the relative $L2$ error and the absolute pointwise error. The formula for the relative $L2$ error is given by:

$$\text{Relative L2 Error} = \frac{\|\hat{u} - u\|_2}{\|u\|_2} \quad (2.14)$$

where \hat{u} is the predicted solution and u is the true solution. The relative $L2$ error measures the discrepancy between the predicted and true solutions, normalised by the $L2$ norm of the true solution. This metric provides a sense of how the error scales relative to the magnitude of the true solution, making it useful for comparing errors across different problems and scales. The formula for the absolute pointwise error is:

$$\text{Absolute Pointwise Error} = |\hat{u} - u| \quad (2.15)$$

The absolute pointwise error quantifies the error at each specific point in the domain by taking the absolute difference between the predicted and true values. This metric is particularly useful for identifying local discrepancies and understanding the performance of the model at specific locations within the domain. Together, these metrics provide a comprehensive understanding of the model's performance, highlighting both overall accuracy and local errors.

2.6 Developments in the PINN frameworks

The PINNs have rapidly evolved, with significant advancements in each of their core components such as sampling strategy, network architecture, activation function etc. These enhancements not only improved the accuracy and efficiency of PINNs but also contribute to the broader field of ML. This section will briefly discuss these developments, highlighting how they address previous limitations of the PINNs.

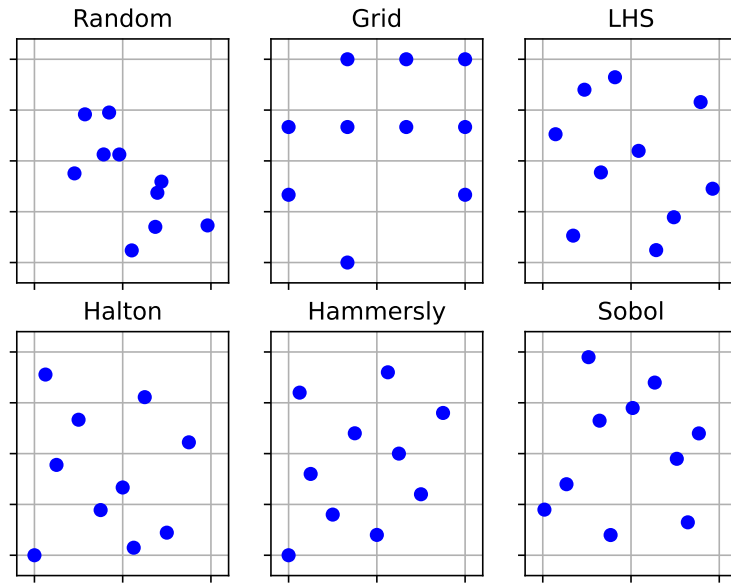


Figure 2.5: The comparison of various sampling methods.

2.6.1 Sampling

Sampling plays a crucial role in the training of PINNs, just as it does in other ML techniques. For the PINNs, this involves generating a point cloud within the domain of interest, which serves as the training data. Various strategies can be employed to sample these points effectively.

A common practice in PINNs is to employ low-discrepancy quasi-random sequences. These sequences are advantageous as they require fewer points than uniformly distributed random points to achieve a comparable level of domain coverage. Essentially, these sequence “spread out” the points in such a way that they are evenly distributed across all the dimensions. Figure 2.5 demonstrates the distribution of 10 points within a unit square for various sampling methods, including random, grid, Latin hypercube sampling (LHS), Sobol, Halton, and Hammersley sequences. The random sampling shows no pattern, which can lead to clustering and gaps. The grid pattern, does not randomise the locations, which may not capture the local variations in the solution. In contrast, quasi-random sequences like the Sobol, Halton, and Hammersley methods provide a more uniform distribution without clustering, which is beneficial for capturing the local variations in the solution [70, 88].

Importance sampling can be seen as substitute of adaptive mesh refinement in PINNs. Rather than using the same sampled points in each training iteration, the points are drawn from a distribution that is proportional to the total loss, \mathcal{L} . Consequently, regions with higher pointwise total loss are sampled more densely, thereby focusing areas where the model needs the most improvement [67, 69, 89].

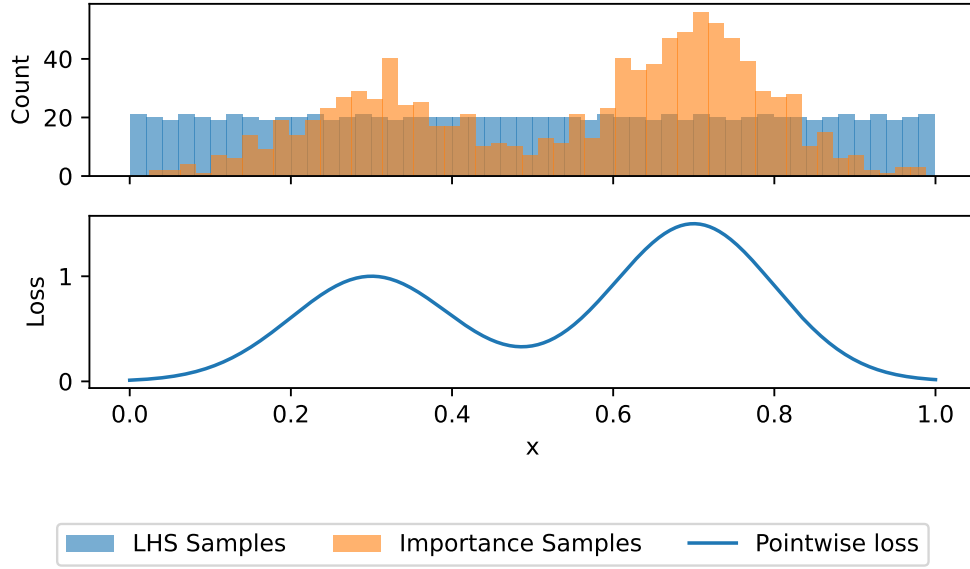


Figure 2.6: The comparison of initial and importance sampling strategies in a 1D feature space x . The top histogram shows initial samples obtained via LHS, and the target distribution of samples derived from importance sampling. The bottom plot shows the pointwise total loss \mathcal{L} across the 1D feature space x , highlighting regions of higher loss where more points are sampled.

Figure 2.6 illustrates the evolution of sampling strategies in a 1D feature space, x , ranging from 0 to 1. Initially, 1000 sample points are distributed uniformly across the feature space using LHS, as depicted by the evenly spaced histogram in blue in the top plot. As the training progresses, samples are drawn from a distribution that aligns with the total loss \mathcal{L} . This distribution, represented by the histogram in orange, is concentrated around regions where the pointwise total loss \mathcal{L} , shown in the bottom plot, is higher. By dynamically adjusting the sampling density in accordance with the pointwise total loss \mathcal{L} , the PINN effectively focuses on learning complex dynamics within the feature space.

2.6.2 Imbalanced loss terms

The individual loss terms in a PINN can exhibit significant differences in magnitude, leading to imbalanced contributions to the total loss. For instance, the \mathcal{L}_{PDE} , which often includes higher-order derivatives, might be substantially lower than the \mathcal{L}_{BC} . This disparity can result in the PINN predominantly learning to satisfy the BCs while ignoring the PDE. Such an imbalance can yield erroneous behaviour, as the problem effectively becomes ill-posed.

A predominant approach to address this issue is the introduction of balancing coefficients for each loss term. These coefficients denoted as λ_{PDE} , λ_{BC} and λ_{IC} are multiplied with the

respective terms in the total loss function (Equation 2.9). By adjusting these coefficients, the relative magnitude of respective loss terms can be balanced [81]. Efforts to automatically adjust these coefficients have led to notable developments, such as self-adaptive PINNs [82] and the self-adaptive weight PINN [83] and the implementation of algorithms like learning rate annealing [71] and neural tangent kernel [84].

2.6.3 Activation function

An activation function, denoted as σ , imparts non-linearity to a NN enabling it to learn complex input-output relationships. Without the activation function, the NN is just a sophisticated linear model with no performance improvement over linear regression. The choice of an appropriate activation function is crucial for convergence in PINNs, as they require smooth activation functions to compute higher-order derivatives present in PDEs. Thus, activation functions with discontinuities, such as the rectified linear unit (ReLU), exponential linear unit (ELU), or scaled exponential linear units (SELU), should generally be avoided because their lack of smoothness introduces inaccuracies in the calculation of higher-order derivatives, which are essential for satisfying the PDE constraints in PINNs. [90].

Mathematically, a NN is a function, where the linear combination of network's weights \mathbf{w} and previous layer's input is passed through the activation function σ which serves as the input to the next layer (Equation 2.16). Jagtap *et al.* [91] proposed the concept of a global adaptive activation function (GAAF), where a trainable parameter A is also passed through the activation function (Equation 2.17). This parameter, acting as the slope of the activation function, allows for more sophisticated feature transformations between the hidden layers. Later, Jagtap *et al.* [92], developed the layer-wise locally adaptive activation functions (L-LAAF), incorporating a distinct trainable parameter, denoted as $A^{(2)}$, in each hidden layer (Equation 2.18). This layer-specific adaptability further enhances the NN's capacity to capture complex behaviours. The standard, GAAF and L-LAAF structures are given as,

$$\hat{\mathbf{u}} = \mathbf{w}^{(3)} \sigma (\mathbf{w}^{(2)} \sigma (\mathbf{w}^{(1)} \mathbf{X})) , \quad (2.16)$$

$$\hat{\mathbf{u}} = \mathbf{w}^{(3)} \sigma (A \mathbf{w}^{(2)} \sigma (A \mathbf{w}^{(1)} \mathbf{X})) \quad (2.17)$$

$$\text{and } \hat{\mathbf{u}} = \mathbf{w}^{(3)} \sigma (A^{(2)} \mathbf{w}^{(2)} \sigma (A^{(1)} \mathbf{w}^{(1)} \mathbf{X})) . \quad (2.18)$$

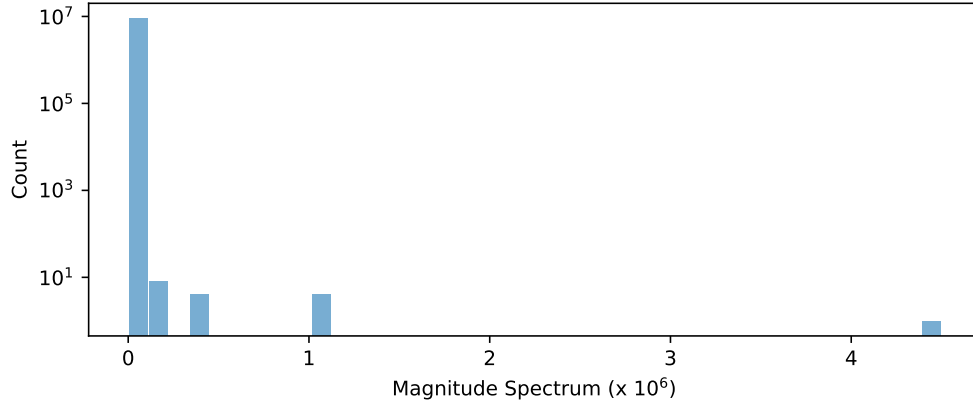


Figure 2.7: Histogram of the magnitude spectrum obtained from the Fourier transform of 2D spatial data, indicating a the presence of high-frequency components due to discontinuities.

2.6.4 Spectral bias in the NN

Spectral bias is a learning bias of NNs towards low-frequency functions. This is a challenge when dealing with high-frequency functions that represent sharp variations, especially in solutions within low-dimensional domains. In Figure 2.7, we present a histogram of the magnitude spectrum derived from a Fourier transform of 2D spatial data exhibiting discontinuities. While the distribution mostly consists of low-frequency components, there are a few high-frequency attributable to the discontinuities. These high-frequency components pose a challenge for traditional FNN architectures, potentially leading to non-convergence issues during training.

2.7 Applications of PINNs to forward problems

In this section, we will briefly discuss the application of PINNs to forward problems. We highlight scenarios where PINNs offer solutions to challenges commonly faced by conventional numerical methods, such as handling parametric problems, complex geometries, and transfer learning.

2.7.1 Parametric problems

The PINNs can be easily extended to solve the problem over a range of parameters, by including them as additional features in the training dataset. These parameters can encompass BCs, IC, coefficients of the PDE and even the geometry of the domain. Consider a training dataset where X represents the input features $[x, y, t]$, for a 2D time-dependent problem, alongside a range of parameters k_i , where i ranges from 1 to n . To learn the parametric solutions, the PINN's training

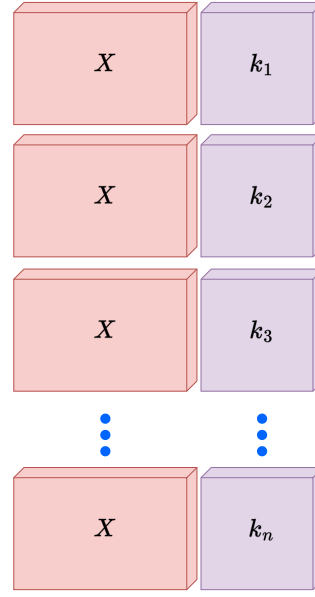


Figure 2.8: Schematic representation of the training dataset with varying parameter k .

dataset is constructed by concatenating the X with each instance of k_i as shown in Figure 2.8 [93]. The extension of the PINN for parametric solutions is illustrated in Figure 2.9.

Recently, Wang *et al.* [94] proposed physics-informed deep operator network (PIDeepONets), an operator learning architecture to solve parametric problems. Similar to PINNs, PIDEeONets only require the PDE, IC and BCs. While a discussion on PIDEeONets is beyond the scope of this chapter, those interested can refer to a survey in Reference [95].

2.7.2 Complex geometry

Jagtap *et al.* [96] proposed the so-called conservative PINN, a space decomposition for the PINNs. This is similar to the concept of elements in FEM, where each element has its own trial function. However, the domain can be decomposed in any arbitrary way without needing any special algorithm as opposed to FEM. Specifically, two additional loss terms were introduced to account for the mismatch in the \mathcal{L}_{PDE} and \hat{u} at the interface of two neighbouring sub-domains.

XPINNs further advanced this by handling space-time domain decomposition (DD) for any irregular geometry [97], effectively handling problems with sharp gradient over complicated geometry. However, this came at the cost of longer training times. To address this, Parallel PINNs were introduced, incorporating efficient parallel algorithms to improve training efficiency and scalability [98]. Hu *et al.* [99] developed theoretical insights on the convergence and generalisation properties of PINNs, enabling accurate modelling of discontinuities, like shockwaves, with prior knowledge of their locations.

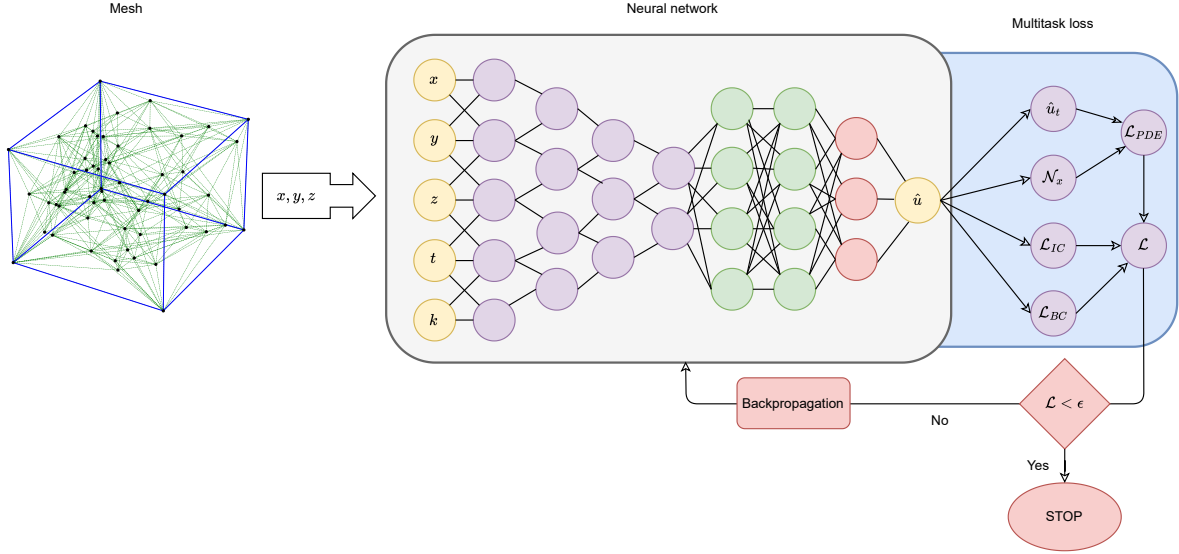


Figure 2.9: The architecture of a PINN a 3D spatio-temporal forward problem with a varying parameter k . Here the inputs are $X = x, y, z, t$ and k , \hat{u} is the predicted output or the solution to the PDE.

2.7.3 Transfer learning

While conventional numerical methods such as multigrid [100] and solution restarting [101] are highly efficient in improving convergence and accuracy, transfer learning in PINNs offers a more straightforward approach. It leverages previously learned solutions to accelerate the solution of new, related problems without requiring the specialised knowledge or complex implementation often associated with traditional methods.

It allows the utilisation of a model trained on one problem, referred to as the base task, to solve similar problem, known as target task. The base task is generally a simpler problem, which may differ from the target task in terms of the geometry, BC or PDE. By utilising a PINN trained on the base task, we can approach more complicated target task, leveraging the pre-trained model, and avoid the lengthy and computationally intensive training from scratch [25, 102].

Figure 2.10, illustrates transfer learning of a pre-trained model to solve the target task with a different geometry and BCs denoted by '*'. We refer the reader to comprehensive overview of transfer learning with PINNs presented in Reference [103].

2.8 Modified baseline PINNs

Numerous modified frameworks of the baseline PINN have been proposed so far. We have specifically chosen PINN frameworks that not only scale to higher dimensions but can also deal

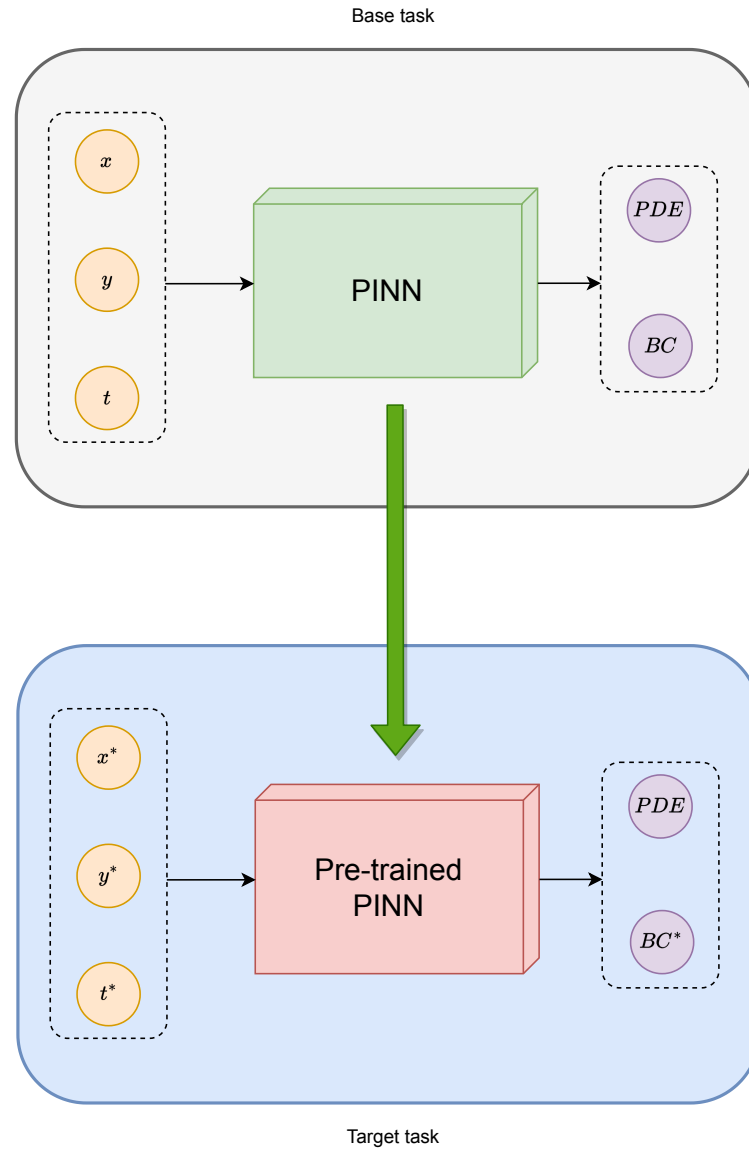


Figure 2.10: Transfer learning from a base task to a target task with different geometry and BC, indicated by '*'.

with spectral bias. One approach to alleviate the spectral bias is to perform input encoding, for example, a transformer [104], to transform the inputs to a higher-dimensional feature space with the help of high-frequency functions.

2.8.1 Fourier Network

In the Fourier network, the input features of a FCNN are encoded into Fourier space using sinusoidal activation functions. Equation 2.19 shows the high-dimensional training dataset, mitigating the

effects of spectral bias.

$$\phi_E = \begin{bmatrix} \sin(2\pi f \mathbf{X}) \\ \cos(2\pi f \mathbf{X}) \end{bmatrix}^T \mathbf{X} \quad (2.19)$$

where $\mathbf{X} \in \mathbb{R}^{d_0}$ is the low dimensional input to network and $\mathbf{f} \in \mathbb{R}^{n_f \times d_0}$ is a trainable frequency matrix. d_0 is the number of features and n_f is the number of frequency sets. These frequencies, similar to network weights, can be sampled from a Gaussian distribution or from a spectral space created from combinations of all entries from a user-defined list of frequencies. The input encoding layer ϕ_E results in a training data with n_f number of features, thus, transforming the input features to a higher dimensional feature space.

2.8.2 Modified Fourier neural network

In a Fourier network (Section 2.8.1), a FCNN is used as the nonlinear mapping between the Fourier features and the model output. The modified Fourier neural network uses a modified version of the FCNN, similar to the one proposed in [71]. The authors were inspired by the neural attention mechanism, which is employed in natural language processing to enhance the hidden states with transformer networks [104].

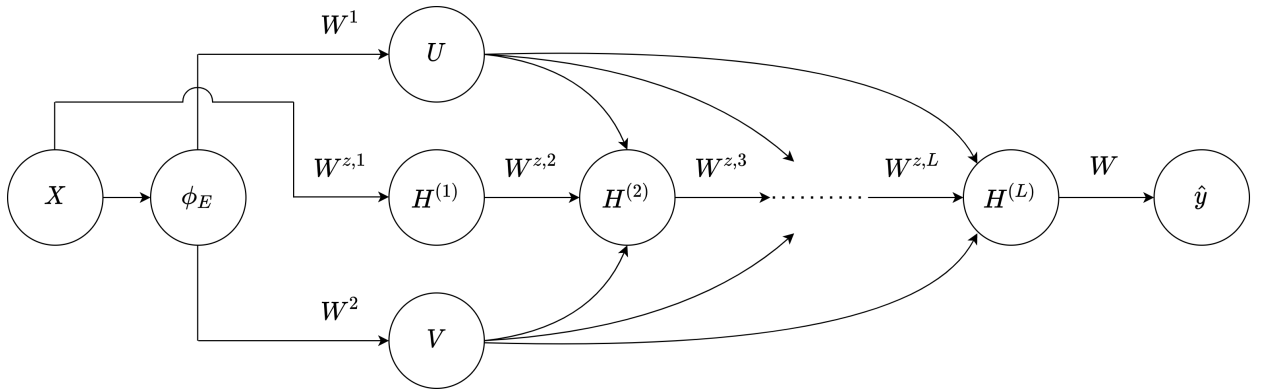


Figure 2.11: Structure of modified Fourier network as per Equation 2.20.

In Equation 2.20, \mathbf{U} and \mathbf{V} are two transformer layers that help in projecting the Fourier features ϕ_E to another feature space, and forward passed through the hidden layers (\mathbf{H}) using the Hadamard product, similar to its standard fully connected counterpart. The Modified Fourier

network takes the following forward propagation rule:

$$\begin{aligned}
 \mathbf{U} &= \sigma(\mathbf{W}^1 \phi_E), \quad \mathbf{V} = \sigma(\mathbf{W}^2 \phi_E) \\
 \mathbf{H}^{(1)} &= \sigma(\mathbf{W}^{z,1} \mathbf{X}) \\
 \mathbf{Z}^{(k)} &= \sigma(\mathbf{W}^{z,k} \mathbf{H}^{z,1}), \quad k = 1, \dots, L \\
 \mathbf{H}^{(k+1)} &= \sigma\left(1 - \mathbf{Z}^{(k)}\right) \odot \mathbf{U} + \mathbf{Z}^{(k)} \odot \mathbf{V}, \quad k = 1, \dots, L \\
 \hat{\mathbf{y}} &= \mathbf{W} \mathbf{H}^{(L+1)}
 \end{aligned} \tag{2.20}$$

where \mathbf{W}^1 , \mathbf{W}^2 , $\mathbf{W}^{z,k}$ and \mathbf{W} are four different sets of weight matrices associated with \mathbf{U} , \mathbf{V} , $\mathbf{H}^{(k)}$ and $\mathbf{H}^{(L+1)}$ with $k = 1, \dots, L$, where L is the number of hidden layers. Figure 2.11 shows the structure of a modified Fourier network.

2.8.3 Sinusoidal Representation Networks (SiReNs)

Sitzmann *et al.* [80] proposed a FCNN which uses the sine trigonometric function as the activation function. This network has some similarities to Fourier networks (Section 2.8.1) as both uses a sine activation function, manifesting the same effect as the input encoding for the first layer of the network.

A key component of this network architecture is the weight initialisation scheme. The weights of the NN are sampled from a uniform distribution $W \sim U(-\sqrt{\frac{6}{f_{in}}}, \sqrt{\frac{6}{f_{in}}})$ where f_{in} is the input size to that layer.

The input of each Sine activation has a Gauss-Normal distribution and the output of each Sine activation, a Sine inverse distribution. This preserves the distribution of activations allowing deep architectures to be constructed and trained effectively.

2.8.3.1 DGM Architecture

The Deep Galerkin method (DGM) architecture [75] (Equation 2.21), consists of several hidden layers, which are referred to as DGM layers, similar to LSTM gates [76], where each layer produces weights based on the last layer, determining how much of the information gets passed to the next layer.

$$\begin{aligned}
\mathbf{S}^{(1)} &= \sigma(\mathbf{W}^1 \mathbf{X}) \\
\mathbf{Z}^{(k)} &= \sigma(\mathbf{V}_z^{(k)} \mathbf{X} + \mathbf{W}_z^{(k)} \mathbf{S}^{(k)}), \quad k = 1, \dots, L \\
\mathbf{G}^{(k)} &= \sigma(\mathbf{V}_g^{(k)} \mathbf{X} + \mathbf{W}_g^{(k)} \mathbf{S}^{(k)}), \quad k = 1, \dots, L \\
\mathbf{R}^{(k)} &= \sigma(\mathbf{V}_r^{(k)} \mathbf{X} + \mathbf{W}_r^{(k)} \mathbf{S}^{(k)}), \quad k = 1, \dots, L \\
\mathbf{H}^{(k)} &= \sigma(\mathbf{V}_h^{(k)} \mathbf{X} + \mathbf{W}_h^{(k)} (\mathbf{S}^{(k)} \odot \mathbf{R}^{(k)})), \quad k = 1, \dots, L \\
\mathbf{S}^{(k+1)} &= (1 - \mathbf{G}^{(k)}) \odot \mathbf{H}^{(k)} + \mathbf{Z}^{(k)} \odot \mathbf{S}^{(k)}, \quad k = 1, \dots, L \\
\hat{\mathbf{y}} &= \mathbf{W} \mathbf{S}^{(L+1)}
\end{aligned} \tag{2.21}$$

The DGM architecture consists of multiple nonlinear transformations of the input: \mathbf{Z} , \mathbf{G} , \mathbf{R} and \mathbf{H} , that helps with learning complicated functions such as discontinuous functions. Figure 2.12 shows the structure of DGM architecture and the DGM layers.

A DGM layer includes \mathbf{Z} , \mathbf{G} , \mathbf{R} and \mathbf{H} with their sets of weights \mathbf{V} and \mathbf{W} . Thus, a DGM layer consists of 8 weight matrices. Additionally, the DGM architecture consists of two more weight matrices: \mathbf{W}^1 and \mathbf{W} .

2.9 Examples

Three test cases are presented: 1D burgers equation, Allen-Cahn equation and Lid driven cavity to showcase the capability of various tools that we have discussed so far. In all the test cases, we used Adam optimiser, with Xavier normal weight initialisation and hyperbolic tangent activation function. The Xavier normal weight initialisation ensures that the weights are initialised to maintain a balance between the variance of inputs and outputs across layers, improving training stability by preventing exploding or vanishing gradients. We sampled the initial set of collocation points with Sobol sequence in all the test cases.

2.9.1 1D Burgers equation

The 1D Burgers equation is a time-dependent problem with details given in Equation 2.22.

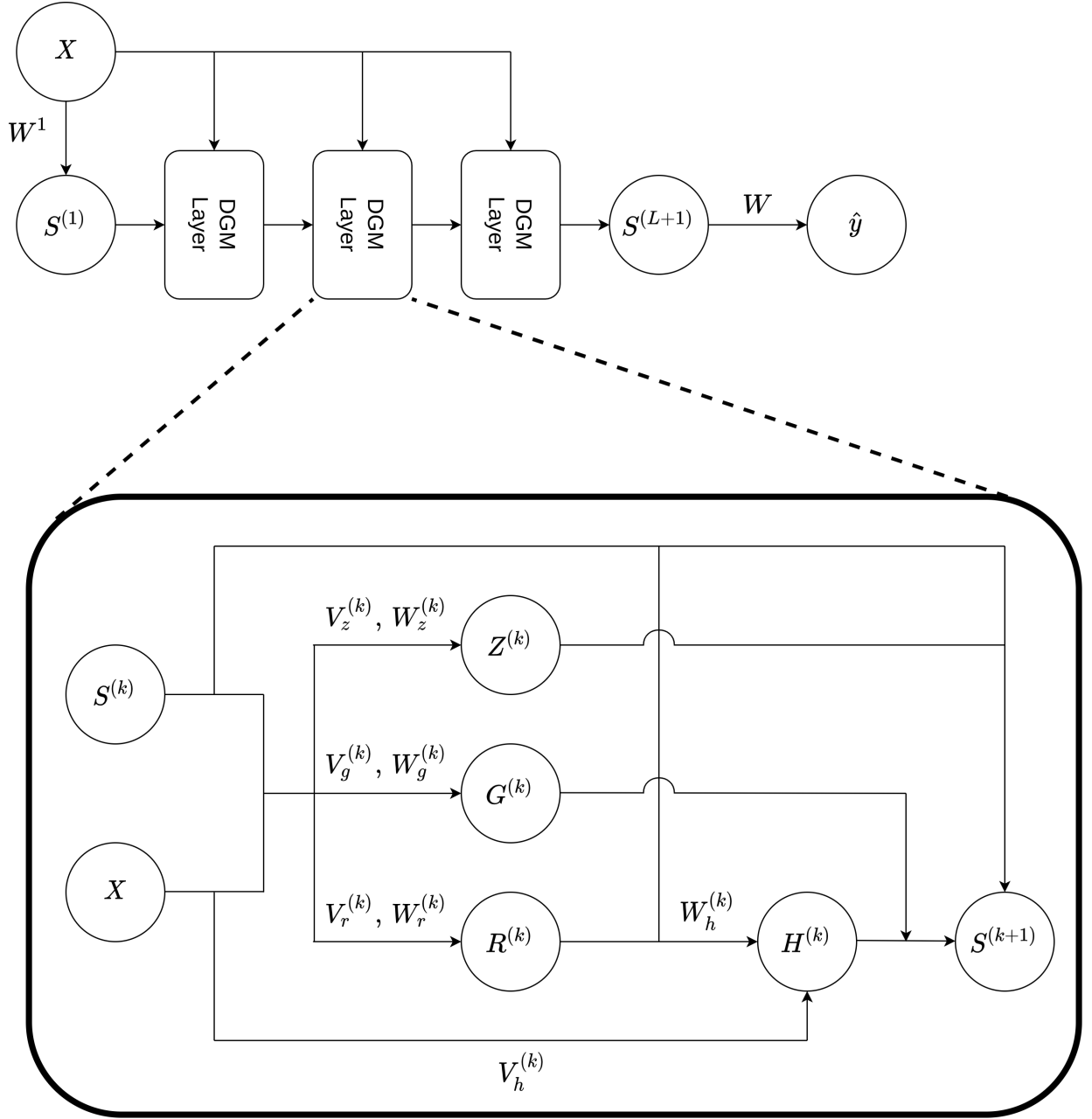


Figure 2.12: Structure of DGM architecture and the DGM layers as per Equation 2.21.

$$\begin{aligned}
 u_t + uu_x - (0.01/\pi)u_{xx} &= 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\
 u(0, x) &= -\sin(\pi x), \\
 u(t, -1) &= u(t, 1) = 0
 \end{aligned} \tag{2.22}$$

The Burgers equation is a second-order non-linear convection-diffusion problem with an

analytical solution available in Reference [19]. The presence of a non-linear convection term uu_x exhibits a discontinuity over time. We sampled points in the spatio-temporal domain and passed them as inputs to the NN (See Figure 2.4). The loss function consist of \mathcal{L}_{PDE} , \mathcal{L}_{BC} , \mathcal{L}_{IC} (See Equation 2.9). These loss terms ensured that the predicted solution satisfied the governing PDE, the BC, and the IC. We employed a FCNN architecture, i.e., a baseline PINN, and trained the model for 10k iterations. Figure 2.13, shows the PINN predicted solution and the absolute pointwise error between the analytical solution and PINN predicted solution.

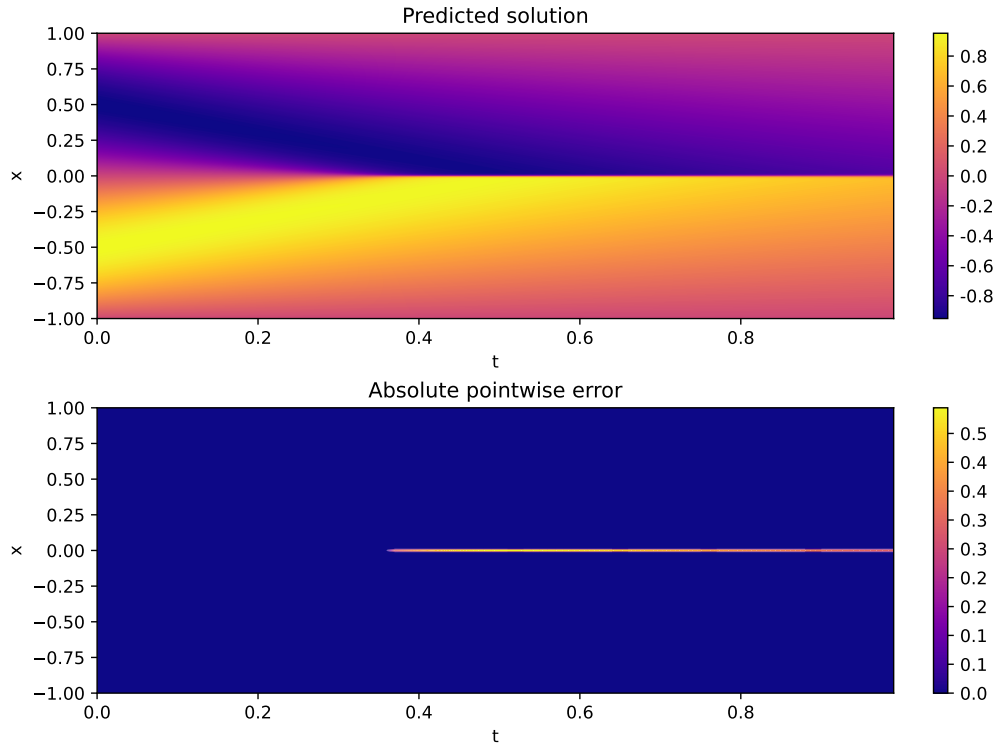


Figure 2.13: Baseline PINN predicted solution of 1D Burgers equation is shown on the top and absolute pointwise error between the analytical solution and PINN predicted solution is shown on the bottom.

The observed discontinuity in the solution may be attributed to the spectral bias inherent to FNNs, which we have previously discussed. Thus, the network is not able to capture the discontinuity as shown in Figure 2.13. This limitation is reflected in the computed relative L2 error of 5.17%.

2.9.2 1D Allen-Cahn equation

The Allen-Cahn equation is used to model the process of phase separation and is characterised by a diffusion term and a non-linear reaction term that drives the system towards minimising its

free energy, often resulting in the creation of interfaces between phases over time. Consider the Allen-Cahn equation along with periodic BC (Equation 2.23).

$$\begin{aligned} u_t - 0.0001u_{xx} + 5u^3 - 5u &= 0, \quad x \in [-1, 1], t \in [0, 1], \\ u(0, x) &= x^2 \cos(\pi x), \\ u(t, -1) &= u(t, 1), \\ u_x(t, -1) &= u_x(t, 1) \end{aligned} \tag{2.23}$$

The ground truth was generated using spectral Fourier discretisation and fourth-order explicit Runge-Kutta time integrator. The solution $u(x, t)$ evolves over time due to the combined effects of diffusion and reaction. The diffusion term $-0.0001u_{xx}$ tends to smooth out variations in u , while the cubic non-linear reaction term $u^3 - 5u$ can create multiple stable states over time.

The initial condition $u(0, x) = x^2 \cos(\pi x)$ is smooth and contains no discontinuities. However, as time progresses, the solution develops sharp transitions between the phases due to the non-linear dynamics, which resembles a stiffness in the numerical solution.

Similar to the 1D Burgers equation the baseline PINN couldn't capture the regions with stiff solution, as shown in Figure 2.14a, leading to a relative L2 error of 1.32%. However, with the application of Fourier NN the PINN was able to accurately capture the stiff regions in the solution, as shown in Figure 2.14b, thus reducing the relative L2 error to 0.06%. This example underscores how employing a high-dimensional training dataset can effectively mitigate the spectral bias.

2.9.3 Lid driven cavity

The lid-driven cavity is a well-known benchmark problem in computational fluid dynamics. It consists of a square cavity with three rigid walls having no-slip conditions and the top lid moving with a tangential unit velocity $u = 1$. The lower left corner of the domain has a reference pressure p of 0 as shown in Figure 2.15.

The Lid driven cavity uses the 2D steady-state incompressible Navier-Stokes equations to model fluid flow, as detailed in Equation 2.24.

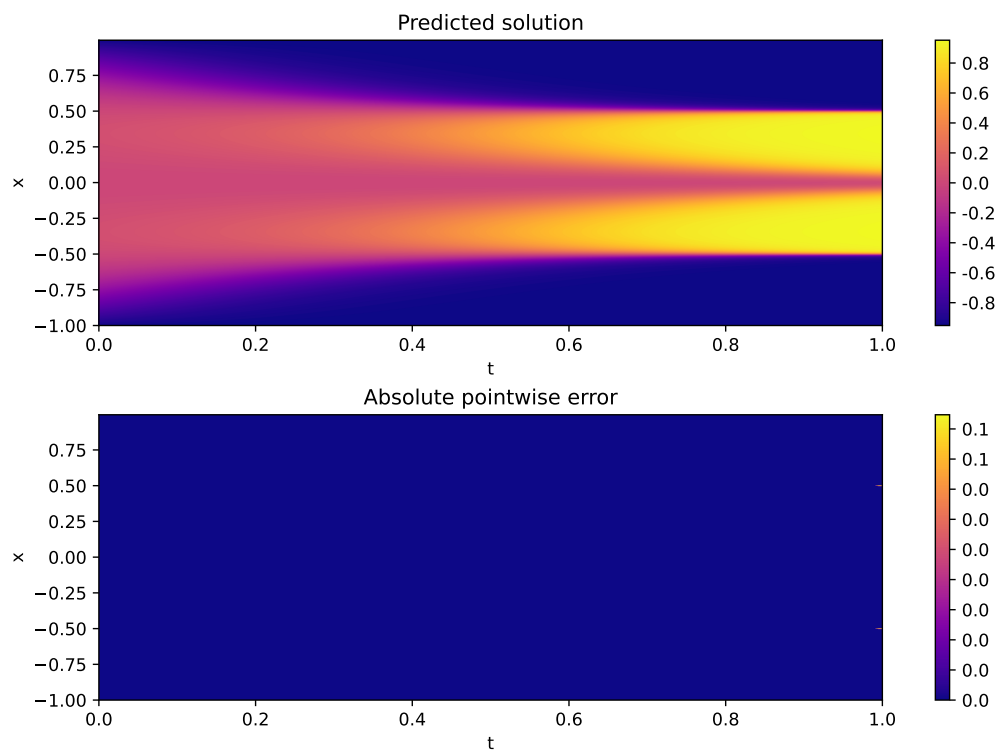
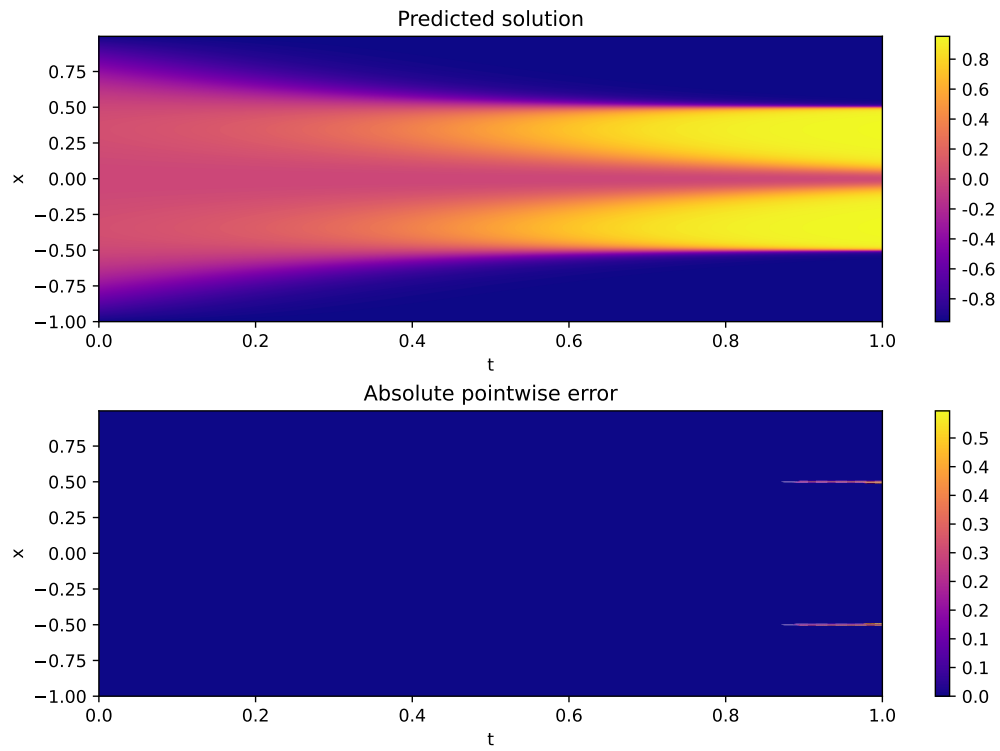


Figure 2.14: Comparison of Baseline PINN and Fourier NN predicted solutions for the 1D Allen-Cahn equation. The top row of each subfigure shows the predicted solutions, while the bottom row displays the absolute pointwise error relative to the analytical solution.

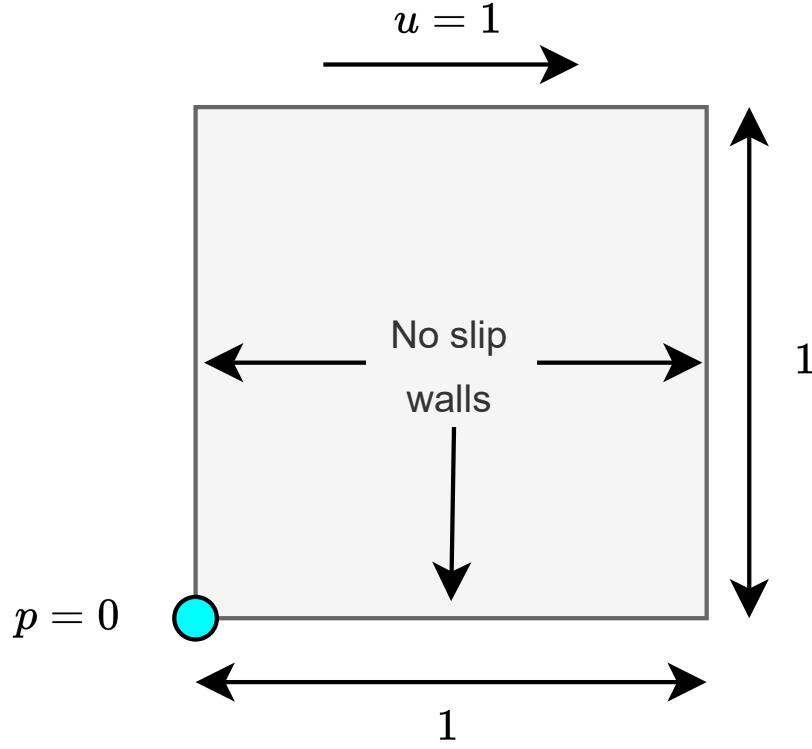


Figure 2.15: Geometry of the lid-driven cavity problem.

$$\begin{aligned}
 \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\
 u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} &= -\frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \\
 u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} &= -\frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right)
 \end{aligned} \tag{2.24}$$

where u and v are velocities in x and y directions respectively, p is the pressure, ν is the kinematic viscosity and ρ is the density of the fluid. Focusing on a Reynolds number of 100 to simulate laminar flow conditions, we set $\rho = 1$ and $\nu = 0.01$. The numerical solution was acquired through the SIMPLE (Semi-implicit method for pressure-linked equation) algorithm [105], and Figure 2.16 shows this solution.

The conflicting BCs on top left and top right corners result in sharp discontinuities. Similar to the Burgers equation, the baseline PINN struggles to accurately capture these discontinuities due to spectral bias. This limitation is shown in Figure 2.17.

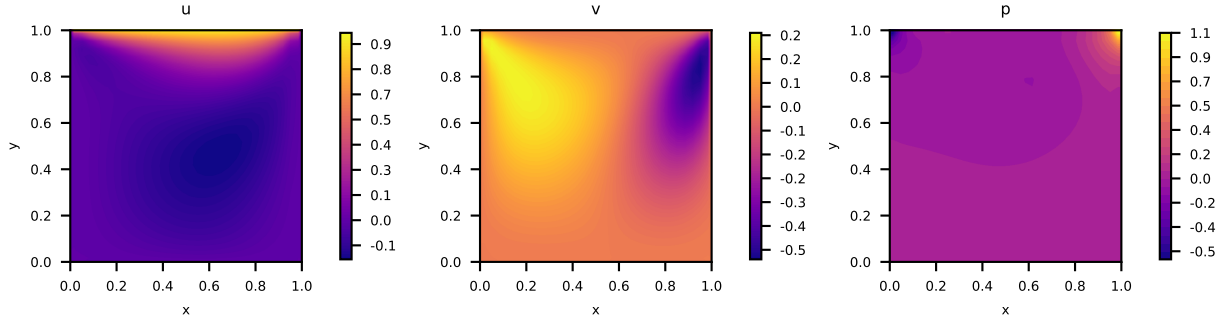


Figure 2.16: Numerical solution for the lid-driven cavity problem, obtained using the SIMPLE algorithm, showcasing the velocity field and pressure distribution.

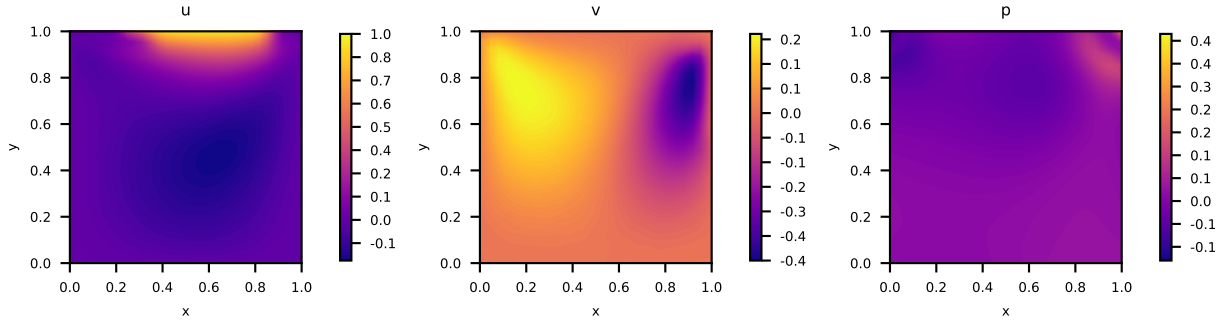


Figure 2.17: Baseline PINN predicted solution of the lid-driven cavity.

To mitigate the issue of spectral bias, we employed the Deep Galerkin Method (DGM) architecture [75], coupled with self-adaptive PINN's weight balancing algorithm, L-LAAF and importance sampling. This approach resulted in a reduction of the relative L2 error by nearly 50% compared to the baseline PINN, as shown in Table 2.2. Figure 2.18, showcases the solution predicted by DGM, highlighting the improved accuracy.

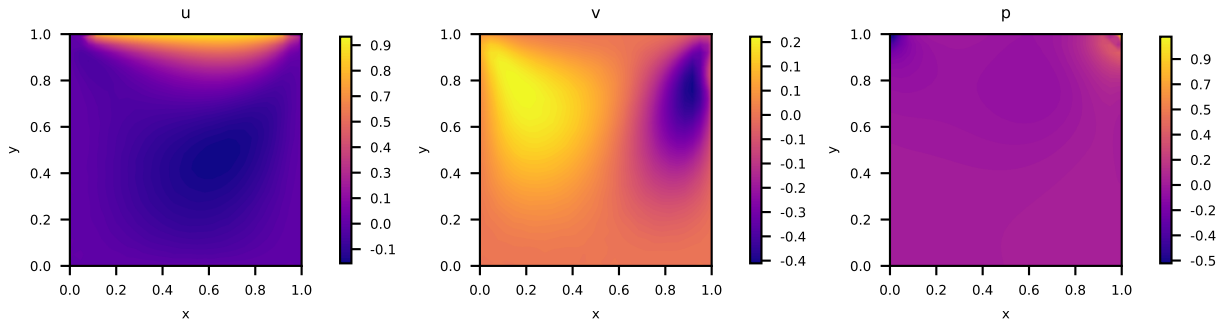


Figure 2.18: DGM predicted solution of the lid-driven cavity.

Following the DGM-based model, we integrated it with the XPINN framework, dividing the

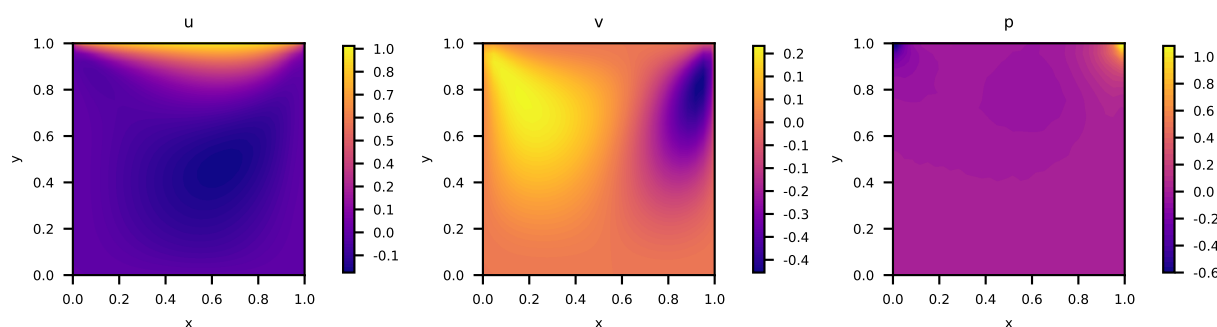


Figure 2.19: DGM predicted solution of the lid-driven cavity with three sub-domains equally spaced along the x -direction.

domain into three equally spaced sub-domains along the x -direction. This division resulted in three times reduction in the relative L2 error compared to the DGM-based model, as detailed in Table 2.2 and predicted solution shown in Figure 2.19. This shows the effectiveness of domain decomposition technique while solving problem involving discontinuities.

Table 2.2: Relative L2 error (in %) for Lid driven cavity.

	u	v	p
Baseline PINN	47.7	26.3	80.4
DGM architecture	26.8	18.3	41.3
DGM with 3 sub-domain	8.6	9.7	12.6

2.10 Conclusions

Physics-informed neural networks have emerged as a powerful framework for solving problems involving PDEs. By seamlessly integrating physical laws into deep learning model, they allow exploring a wide range of scientific and engineering challenges. This chapter has provided a comprehensive overview of PINNs, covering the core components, advancements in sampling strategies, multitask loss function challenges, adaptive activation functions, and the issues with spectral bias, enhanced accuracy and efficiency of PINNs. Notably, PINNs offer significant advantages over traditional numerical techniques, enabling:

- The application of transfer learning to leverage insights from solved problems on new and similar challenges.
- Efficiently addressing high-dimensional parametric problems.

- Simplified domain decomposition for stiff problems, avoiding complex algorithms required for mesh generation.

We solved the 1D Burgers equation, 1D Allen-Cahn equation and the lid-driven cavity problem to showcase the various improvements over the baseline PINN that allowed for effective handling of discontinuities. These improvements mark PINNs as a powerful tool for efficiently solving complex problems. As we enhance PINNs further, we believe their full potential is yet to be realised, promising more innovative solutions in the future.

Chapter 3

Solving stiff-PDEs using PINNs

In this chapter, we have investigated stiff-PDEs [106], specifically those with a discontinuous solution, such as conflicting BCs at adjacent edges or faces. The reason for this focus is due to the significant challenge such problems pose with PINNs because it is not possible to fit a smooth curve such as a NN on these discontinuities. We leverage PINN-based open-source frameworks to solve these problems and compare their capabilities. We also introduced tools specifically designed for solving discontinuous problems.

Libraries such as DeepXDE [107], SciANN [108], TensorDiffEq [109] and NeuralPDE [110] are easy to implement and can be used to solve simple problems in the 1D and 2D spatial domains. NVIDIA Modulus (formerly NVIDIA SimNet) is an advanced open-source library providing access to different PINN frameworks. NVIDIA Modulus has been successfully used to simulated physical systems such as laminar flow over a Field Programmable Gate Arrays (FPGA) heatsink, turbulent flow over a simple 3D heatsink with parameterised fin dimensions, conjugate heat transfer over NVIDIA NVSwitch heat sink using transfer learning [68, 79].

We took two heat conduction problems (2D and 3D) with a discontinuous solution at corner points as test cases. In Sections 3.4 and 3.5, we investigated these problems with a number of PINN frameworks from Table 3.1 and compared the results with the FEM solution. PINNs are also known for solving parametric PDEs [93, 111–113]. In Section 3.6, we investigated the 2D test case with parameterised conductivity and geometry.

For the forward problems, we used the baseline PINN, DeepXDE’s baseline PINN, NVIDIA Modulus’s baseline PINN, Fourier network, Modified Fourier network, SiReNs and DGM architecture. Whereas for the parametric PDE, we have only used the DeepXDE’s baseline PINN, Modified Fourier network and DGM architecture.

3.1 Challenges in Solving Stiff-PDEs with PINNs

3.1.1 Exact BC Imposition

The output of the PINN can be hard constrained to exactly satisfy the BCs [114]. A function is manually constructed to transform the network outputs to exactly satisfy the BCs. Generally, hard constrained BCs work with simple geometry, constant individual Dirichlet BCs without conflicting each other. For example, given two BCs $u(0) = 0$ and $u(1) = 10$, the output layer can be transformed using the function $\hat{u} := \hat{u}(x)(x - 1) + 10x$, where \hat{u} is the original network output. Generally, hard-constrained BCs work with simple geometries and BCs. This approach does not work well with stiff PDEs, as noted by Sukumar *et al.* [115]. In Section 3.4.4, we discussed the difficulties with the exact imposition of BCs in stiff PDEs with discontinuous solutions.

3.1.2 Overfitted vs generalised solution

The baseline PINN gained its popularity due to the notion that it can solve PDEs with sparse sampling in the spatio-temporal domain. Later on it was realised that, in the case of stiff-PDEs one must sample enough data points to capture the local variations in the solution [71].

Theoretically, one can obtain an overfitted model as well as a generalised model depending on the number of data points in the training dataset. An overfitted model exhibits low training loss but high validation and testing loss whereas a generalised model exhibits low training, validation and testing error. Thus overfitted model can only be used for inferring the solution from the training dataset, i.e. any data point of interest should be included in the training dataset. On the other hand, a generalised model can be obtained by sampling a significantly greater number of data points in the spatio-temporal domain. The solution can be predicted at new spatio-temporal locations within the domain.

The overfitted or generalised model is a qualitative aspect that depends on human judgement. In the case of stiff-PDEs, the overfitted model may converge with a considerable amount of pointwise error. Thus, it is important to sample more points around the boundary or the problematic region. In this thesis, to be on the safer side, we have aimed for a generalised solution, i.e. we sampled a large number of points. Specific details can be found in Section 3.4.5.

3.2 Tools for Solving Stiff-PDEs with PINNs

In this section, we have discussed tools that enhance the accuracy and efficiency of PINNs. Outside the scope of this thesis are tools such as gradient enhanced training [116], learning rate annealing [71], neural tangent kernel [84], integral continuity planes [68]. The gradient enhanced training does not always improve the results compared to the baseline PINN. They may even adversely affect the training convergence and accuracy [117]. The learning rate annealing and neural tangent kernel deals with imbalanced losses where there is a significant difference between the magnitude of individual losses, which is not applicable to our test cases. The integral continuity planes are only applicable to problems involving fluid flow, where the mass flow rate or the volumetric flow rate is applied as an additional constraint, if known. This is particularly useful in the case of channel flow.

3.2.1 Signed distance function

While there have been several efforts to solve stiff-PDEs with discontinuity inside the spatial domain, so far, there is only one viable technique to solve stiff-PDEs with conflicting BCs at the adjacent edges and corners. The difficulty lies in the fact that the activation functions are smooth, i.e. they are differentiable and can't capture discontinuous BCs. Thus, currently the only way to alleviate the issue is to exclude the points with a discontinuity from the training data set.

In DeepXDE, these corner points are excluded by default, as the normal vector at those corners are not defined, or in other words the derivative at those points are not defined, so any BC that includes a derivative for example, a Neumann BC is not defined at the corners.

NVIDIA Modulus and several other literature takes this one step further by using signed distance function (SDF) weights [115, 118, 119]. SDF weights are used to assign minuscule weights around the region with conflicting BCs. This way a region with a discontinuous solution gets a lower priority compared to the regions where the solution is smooth. The application of SDF weights on complex geometry is an active field of research.

3.2.2 Importance sampling

Nabian *et al.* [67] proposed a sampling strategy for efficient training of PINNs based on an approximation method called importance sampling [89], which is often used in reinforcement learning for approximating the expected reward based on the older policy [120, 121]. In optimisation,

the optimal parameters θ^* is defined such that

$$\begin{aligned}\theta^* &= \operatorname{argmin}_{\theta} \mathbb{E}_f [\mathcal{L}(\theta)] \\ &\approx \operatorname{argmin}_{\theta} \frac{1}{N} \sum_{j=1}^N \mathcal{L}(\theta; \mathbf{x}_j), \quad \mathbf{x}_j \sim f(\mathbf{x}),\end{aligned}\tag{3.1}$$

where $\mathbb{E}_f [\mathcal{L}(\theta)]$ is the expected value of the total loss \mathcal{L} , when the collocation points are sampled from f the sampling distribution in the physical domain $\mathbf{x}_j \in \Omega$.

Typically, we use a uniform distribution for sampling the collocation points. In importance sampling, the collocation points are drawn from an alternative sampling distribution $q(x)$, and the NN parameters are approximated as per Equation 3.2 instead of Equations 2.5 and 2.6.

$$\theta^* \approx \operatorname{argmin}_{\theta} \frac{1}{N} \sum_{j=1}^N \frac{f(\mathbf{x}_j)}{q(\mathbf{x}_j)} \mathcal{L}(\theta; \mathbf{x}_j), \quad \mathbf{x}_j \sim q(\mathbf{x}).\tag{3.2}$$

Sampling the collocation points in each epoch according to $q(x)$ (Equation 3.3), i.e. a distribution proportional to the loss function \mathcal{L} improves the efficiency of PINNs without introducing a hyperparameter.

$$q_j^{(i)} \approx \frac{\mathcal{L}_j^{(i)}}{\sum_{j=1}^N \mathcal{L}_j^{(i)}}, \quad \forall j \in 1, \dots, N\tag{3.3}$$

where $\mathcal{L}_j^{(i)}$ is the total loss of the j th sample in the training dataset in i th epoch. Areas with higher $q^{(i)}$ are sampled more frequently in the i th epoch.

3.3 Open-source libraries

DeepXDE and NVIDIA Modulus are two prominent libraries for using PINN that involves tools to solve stiff-PDEs with discontinuous solutions.

3.3.1 DeepXDE

DeepXDE, originally named SciCoNet (Scientific Computing Neural Networks), is a popular Python-based physics-informed machine learning library for solving forward and inverse problems involving PDEs. The library supports multiple deep learning backends, including PyTorch [122],

TensorFlow [123], JAX [124], PaddlePaddle [125] and MXNet [126]. DeepXDE features its own version of the baseline PINN, that not only improves the accuracy, but helps in faster convergence as discussed in Section 3.2.1.

Other than baseline PINNs, DeepXDE can also solve forward and inverse integro-differential equations (IDEs) [127], fractional PDEs (fPDEs) [127], stochastic PDEs (sPDEs) [128], topology optimization with hard constraints (hPINN) [27], PINN with multi-scale Fourier features [94] and multifidelity neural network (MFNN) [129, 130].

DeepXDE also features NNs for nonlinear operator learning such as DeepONet [131], POD-DeepONet [132], MIONet [133], DeepM&Mnet [134, 135] and multifidelity DeepONet [136].

It supports tensor libraries such as TensorFlow, PyTorch, JAX, and PaddlePaddle. There are two notable features in DeepXDE. The first one is that it chooses the model with least training loss. The second one is that it does not include those corner points no matter what the problem is, with the justification that normal vectors are not defined at those corners to be able to apply the Neumann BCs.

3.3.2 NVIDIA Modulus

NVIDIA Modulus, formerly known as NVIDIA SimNet, is an advanced physics-informed machine learning package. The version 22.03 of NVIDIA Modulus was rewritten from TensorFlow 1.15 to PyTorch due to its ease of use. Although it was originally open-source, NVIDIA's license imposed many restrictions. However, version 23.05 was released under the Apache License, allowing users to freely use the product. NVIDIA Modulus 24.04 is quite sophisticated and provides tools for climate prediction. Nevertheless, we will use version 22.03, as it was the latest version available when this work was conducted.

NVIDIA Modulus employs the integral formulation of loss to normalise the magnitude of loss terms for point clouds with different densities in a geometry (Section 2.5.3). It also uses SDF weights to avoid problematic edges and corners (Section 3.2.1). The SDF weights result in better convergence and improved accuracy. NVIDIA Modulus employs the SDF weights on the collocation points (they refer to these as interior points) and the boundary points separately. We have discussed the variation of SDF weights in the spatial domain in Sections 3.4 and 3.5.

NVIDIA Modulus features forward models such as Fourier networks [78], Modified Fourier networks [68, 79], Sinusoidal Representation networks (SiReNs) [80], Highway Fourier network [137], multi-scale Fourier feature network [94], spatial-temporal Fourier Feature network [131], DGM architecture [75] and multiplicative filter network [68].

NVIDIA Modulus also features NNs for nonlinear operator learning such as Fourier neural

operator (FNO) [137], adaptive Fourier neural operator (AFNO) [138], Physics-informed neural operator (PINO) [139], DeepONet [131], Pix2Pix net [140] [141] and super-resolution net [142].

3.4 Problem 1: 2D steady-state heat conduction

For problem 1, we chose a 2D heat conduction problem with conflicting BCs at the corners. We solved the problem with different PINN frameworks and compared with the FEM solution. The 2D heat conduction problem can be stated as follows:

$$\begin{aligned} u_{xx} + u_{yy} &= 0, & x \in [-0.5, 0.5], & y \in [0, 1] \\ u(x, 0) = u(x, 1) &= 0 & u(-0.5, y) = u(0.5, y) &= 1 \end{aligned} \quad (3.4)$$

We used MATLAB Partial Differential Equation Toolbox [143] to solve the 2D steady-state heat conduction problem (Equation 3.4) with quadratic triangles (Figure 3.1).

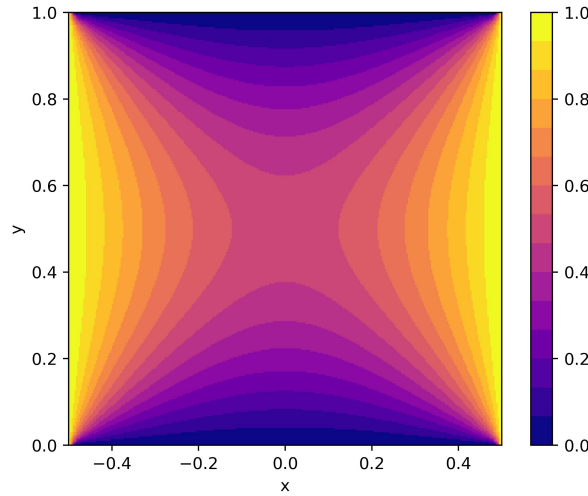


Figure 3.1: FEM solution of 2D steady-state heat conduction problem (Equation 3.4) using MATLAB Partial Differential Equation Toolbox.

We assigned a model number to each of the PINN frameworks (see Table 3.1). Table 3.2 summarises the network parameters and relative L^2 error for various PINN frameworks. Figures 3.4 and 3.2 show the solution of 2D steady-state heat conduction problem for different PINN frameworks.

We have not completed a full exploration of the best parameters for each PINN such as the

number of hidden layers, the number of neurons in each layers, number of boundary and collocation points to be sampled etc. to use in these models. We started with the default values and retained the results if they were acceptable. We observed early convergence in several models, so we stopped training those models. Unless otherwise specified, the same applies to other upcoming problems. As the PINN reaches maturity we will hopefully come up with the best practices to benchmark different PINN frameworks.

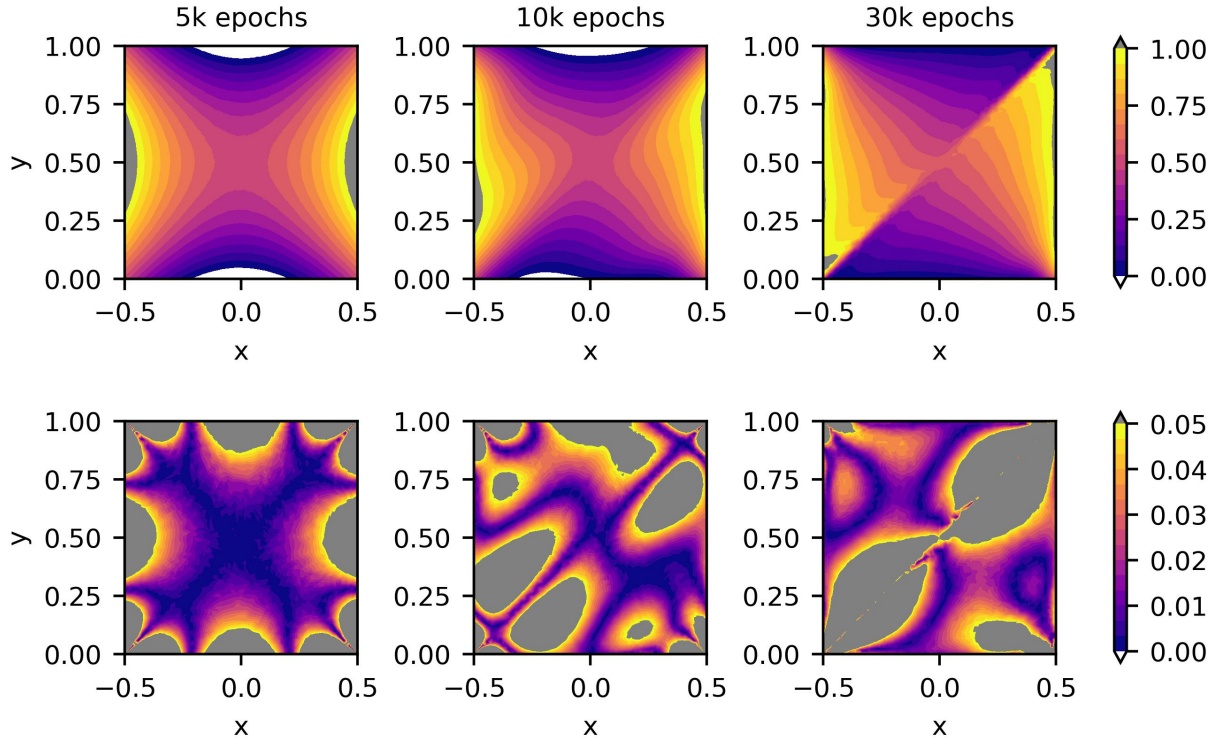


Figure 3.2: Solution of 2D steady-state heat conduction problem using Model 1. The predicted temperature distribution (on the top) and the absolute pointwise error (on the bottom) is shown at 5k, 10k and 30k epochs respectively. Temperature predicted outside the expected bound i.e. when $u \notin [0, 1]$, is shown using the white and grey colour.

3.4.1 Model 1: baseline PINN

Firstly, we trained a baseline PINN with 8 hidden layers and 20 neurons in each layer using the L-BFGS optimiser with hyperbolic tangent activation. We used the nodal coordinates from the mesh of FEM solution consisting of 612 boundary points and 5800 collocation points. We observed that the gradient explodes while training with L-BFGS. That is why we switched to the Adam optimiser with default parameters, unless otherwise stated.

Table 3.1: Models considered in the numerical study

Model number	Model name
Model 1	Baseline PINN
Model 2	DeepXDE's baseline PINN
Model 3	NVIDIA Modulus's baseline PINN (no SDF weights)
Model 4	NVIDIA Modulus's baseline PINN (Interior SDF weights)
Model 5	NVIDIA Modulus's baseline PINN (Full SDF weights)
Model 6	NVIDIA Modulus's baseline PINN (Full SDF weights with importance sampling)
Model 7	NVIDIA Modulus's baseline PINN (Full SDF weights with adaptive activation, importance sampling and quasi-random sampling*)
Model 8	Fourier network (Full SDF weights with adaptive activation, importance sampling and quasi-random sampling*)
Model 9	SiReNs network (Full SDF weights with importance sampling and quasi-random sampling*)
Model 10	Modified Fourier network (Full SDF weights with adaptive activation, importance sampling and quasi-random sampling*)
Model 11	DGM architecture (Full SDF weights with adaptive activation, importance sampling and quasi-random sampling*)

* Sampling drawn from a Halton sequence

Table 3.2: Problem 1: Summary of training parameters and relative L^2 error for different PINN frameworks.

	Layers	Nodes	Boundary points	Collocation points	α	Epochs	Relative L^2 error
Model 1	8	20	612	5800 *	1e-3	30k	0.13599
Model 2	8	20	800	2500 **	1e-3	30k	0.05285 #
Model 3	6	512	4000	4000	1e-3	20k	0.08931
Model 4	6	512	4000	4000	1e-3	10k	0.10124
Model 5	6	512	4000	4000	1e-3	10k	0.09557
Model 6	6	512	4000	4000	1e-3	20k	0.06448
Model 7	6	512	4000	4000	1e-3	20k	0.06433
Model 8	6	512	4000	4000	1e-3	20k	0.18741
Model 9	6	512	4000	4000	2e-5	11k	0.12640
Model 10	6	512	4000	4000	1e-3	20k	0.17577
Model 11	6	512	4000	4000	1e-3	30k	0.08229

* Model 1 uses the nodal coordinates from the FEM mesh.

** Model 2 uses the Sobol sequence.

Relative L^2 error not calculated at corner points.

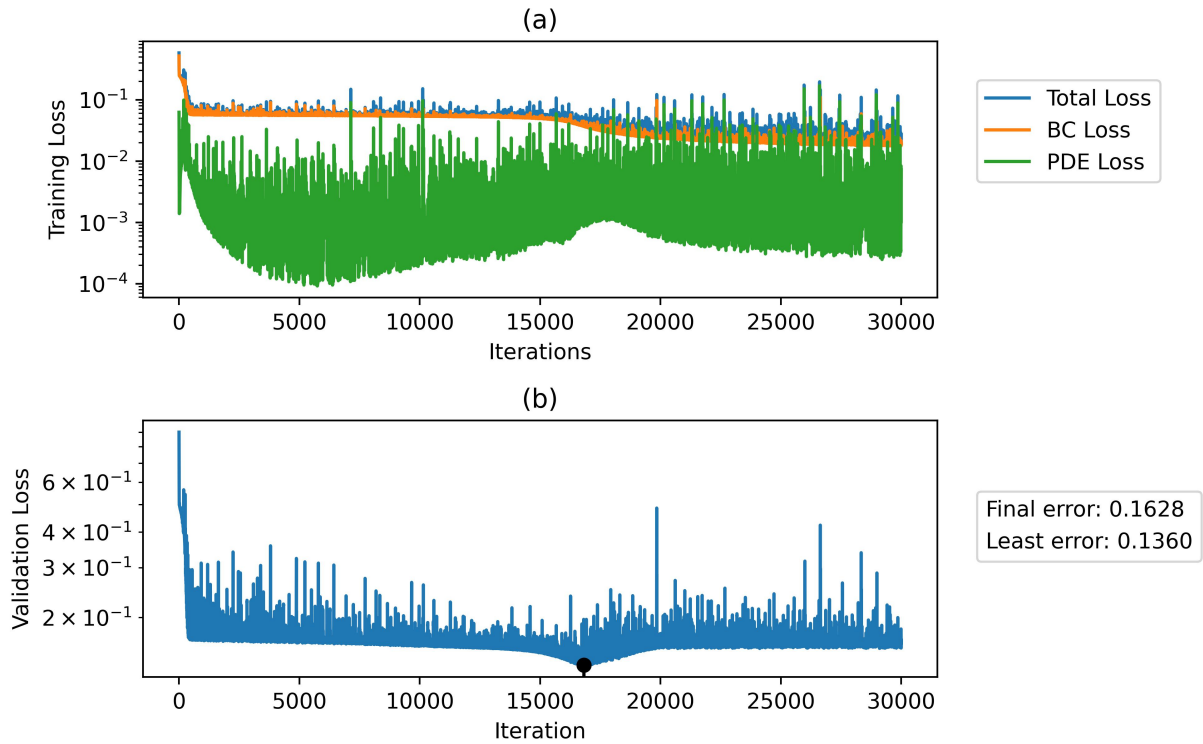
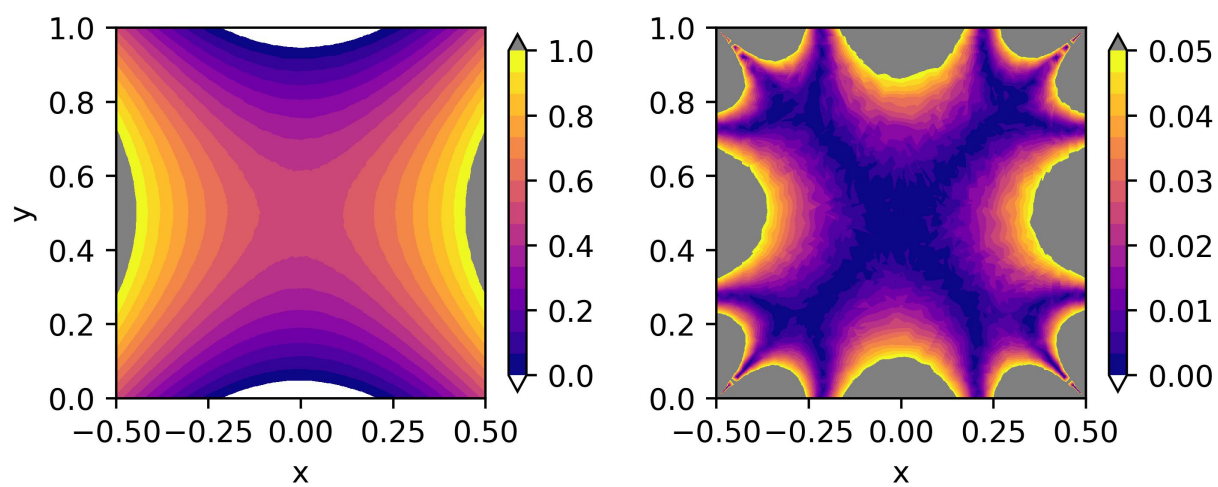


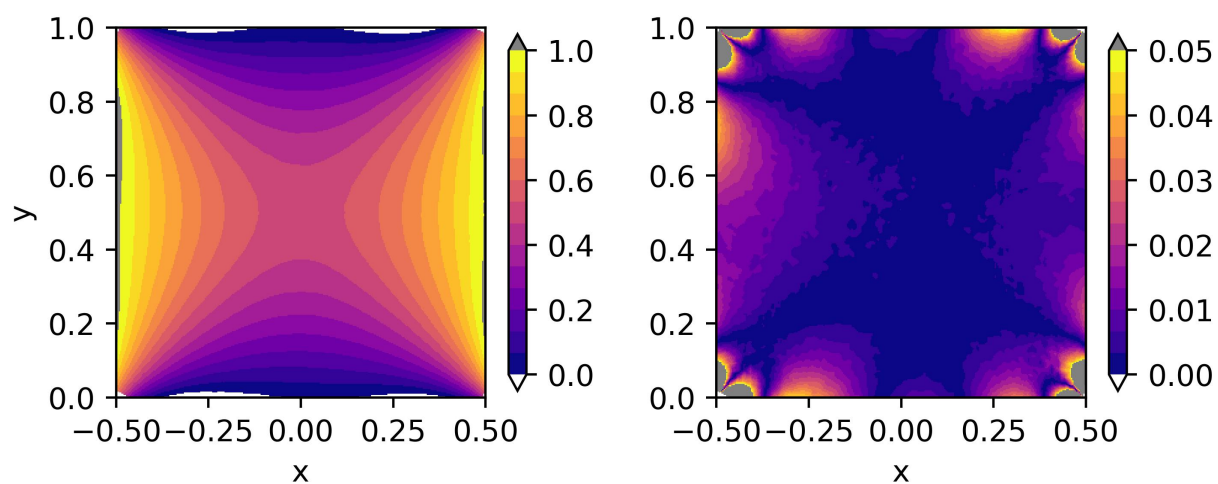
Figure 3.3: The training loss (top) and the validation loss (bottom) for the 2D steady-state heat conduction problem. The training loss plot shows the BC loss (BC loss), the residual loss (PDE loss) and the sum of both, i.e. the total loss. The validation loss plot shows the mean squared error between the temperature predicted from the training dataset and the FEM solution. The black dot denotes the least validation loss.

Figure 3.2, shows the predicted temperature distribution and pointwise absolute error at 5k, 10k and 30k epochs. Initially, i.e. around 5k epochs, the pointwise absolute error around the corners is much higher compared to the interior points. If we continue the training past 5k epochs, the optimiser tries to reduce the loss around the corners at the cost of high errors at the interior points, which can be clearly seen in the pointwise absolute error plot at 30k epochs. This is a result of the fact that we did not use a learning rate scheduler [144]. A learning rate scheduler adjusts the learning rate α between epochs during the training which helps convergence.

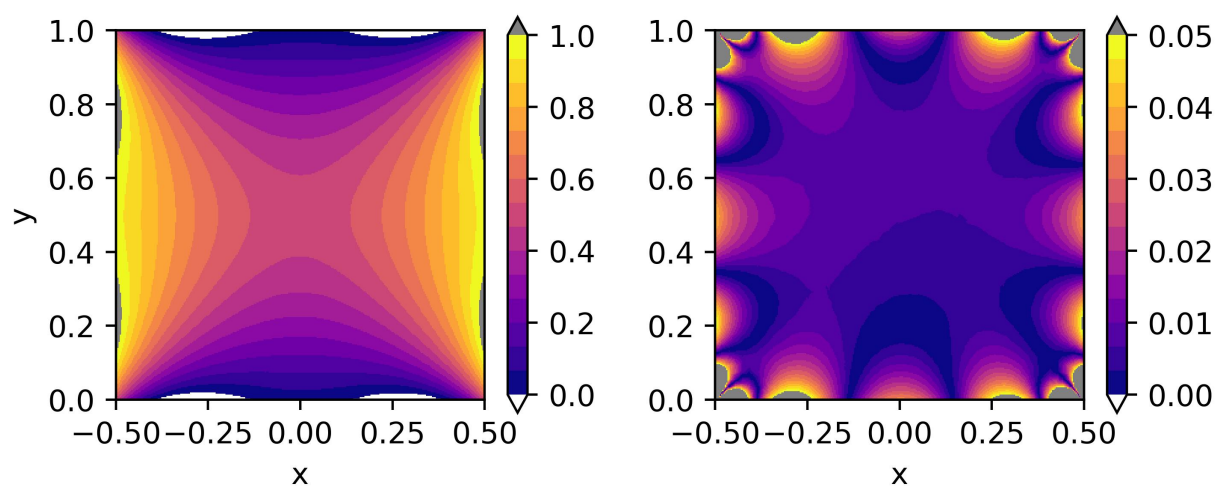
Figure 3.3, shows the training and validation losses. We purposely used the predicted temperature distribution from the training dataset against the FEM solution to calculate the validation loss. In the validation loss plot, we can clearly see that the least validation error occurs somewhere between 15k and 20k. However, the total training loss (orange colour) continues to decrease even after 20k epochs resulting in increased validation loss in the interior points as discussed earlier.



(a) Model 1

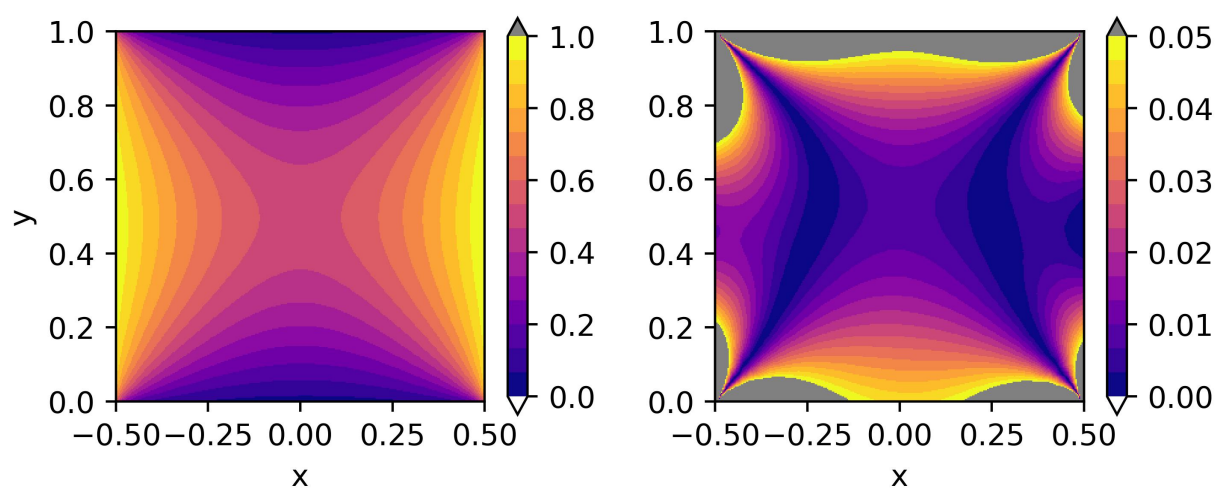


(b) Model 2

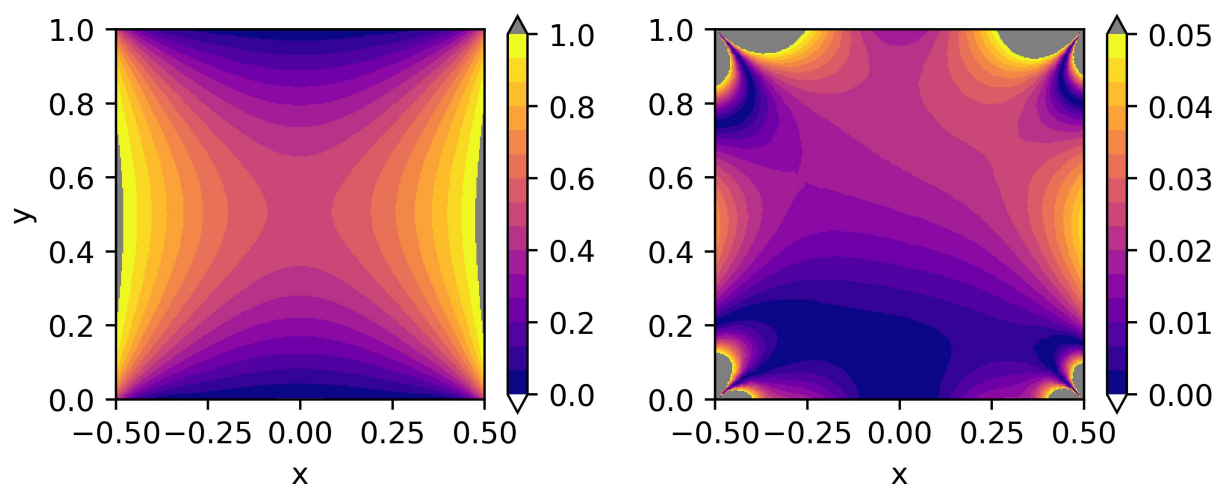


(c) Model 3

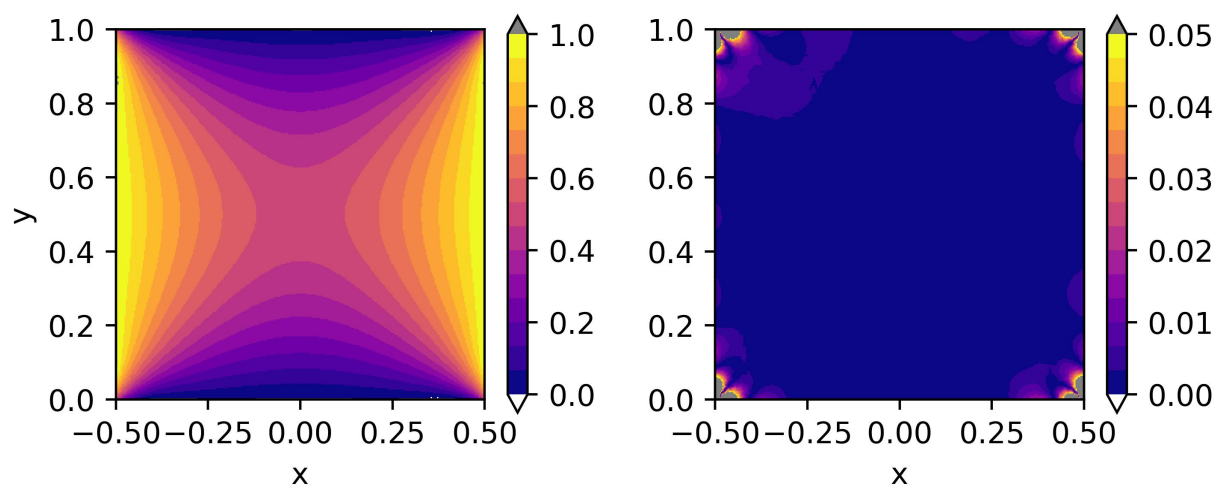
Figure 3.4: Continued on next page.



(d) Model 4

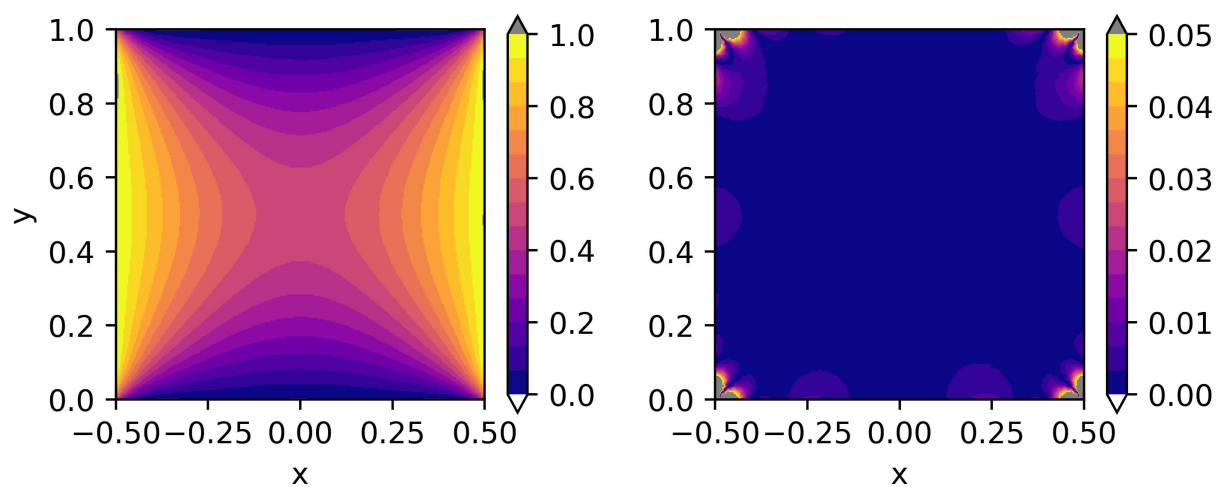


(e) Model 5

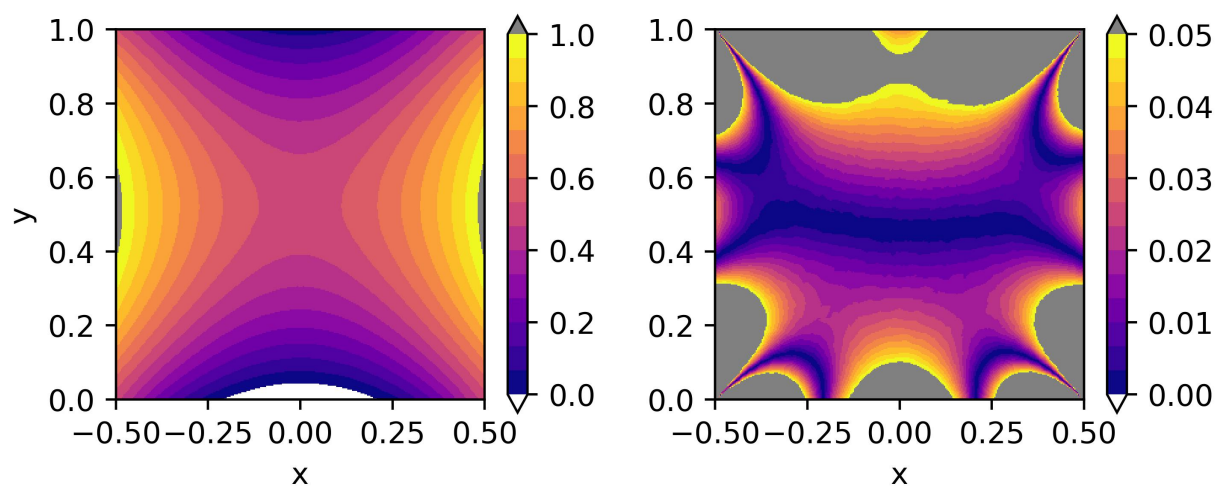


(f) Model 6

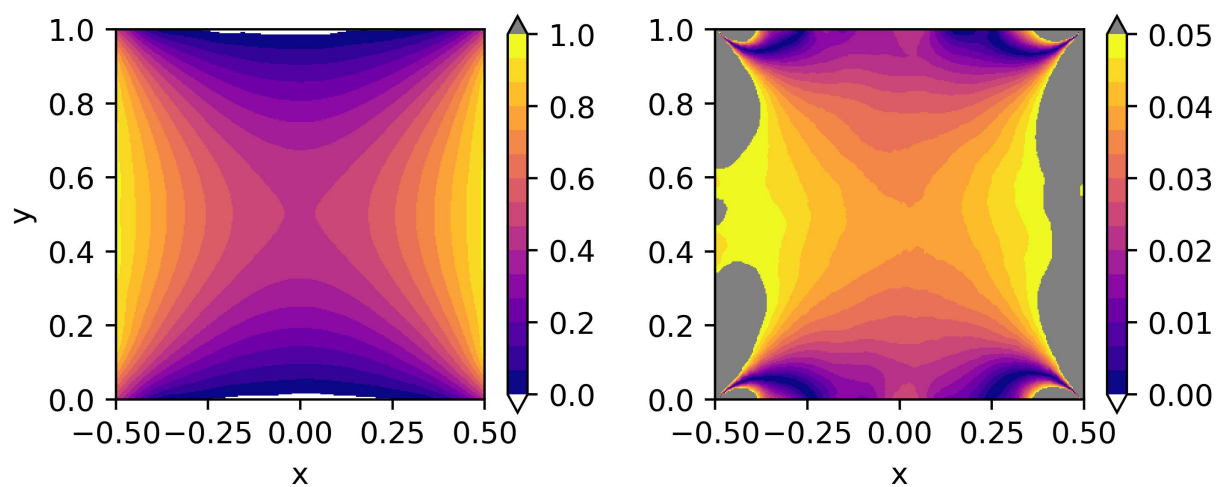
Figure 3.4: Continued on next page.



(g) Model 7

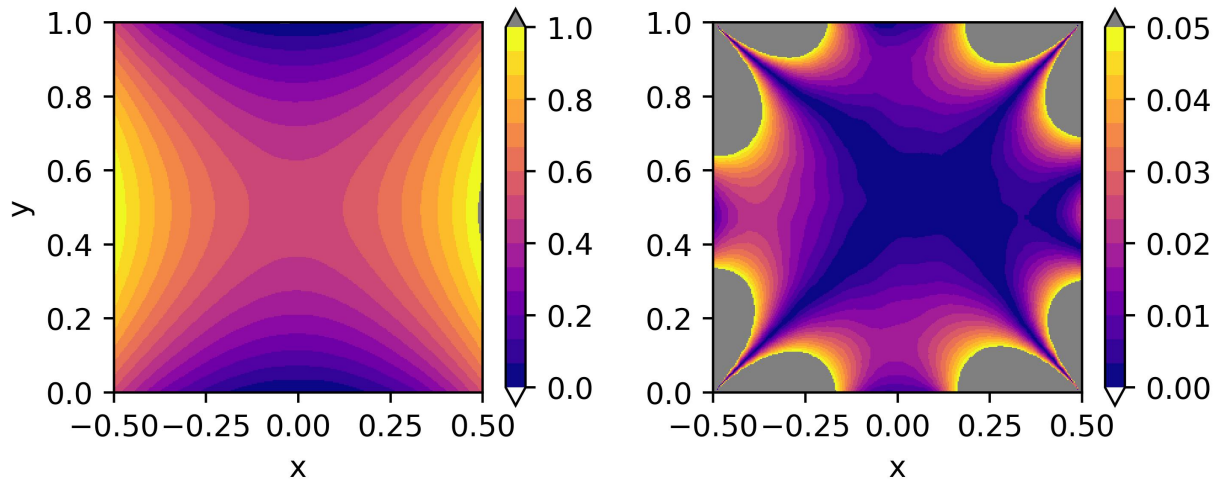


(h) Model 8

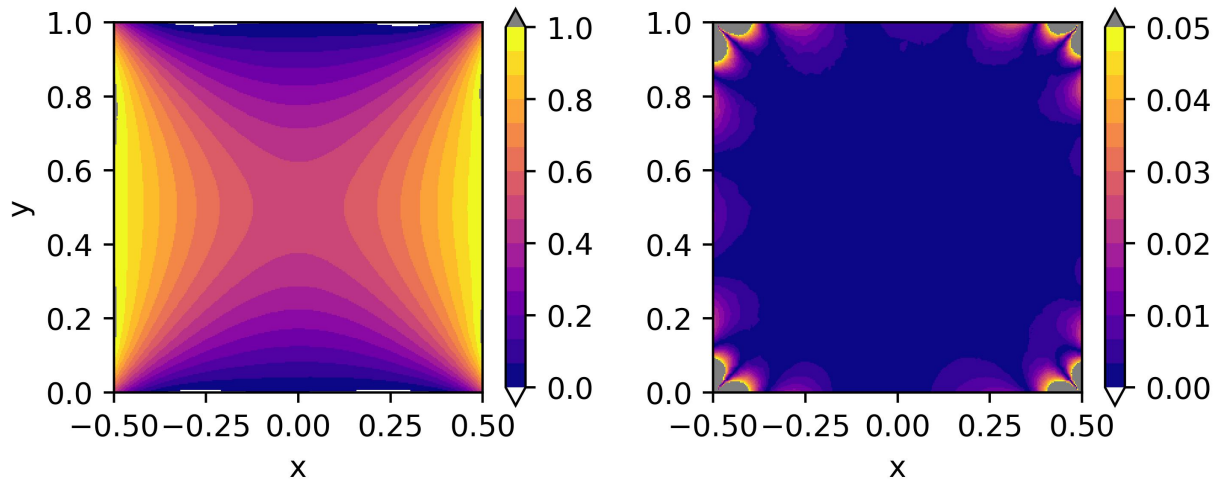


(i) Model 9

Figure 3.4: Continued on next page.



(j) Model 10



(k) Model 11

Figure 3.4: PINN predicted solution of 2D steady-state heat conduction problem is shown on the left side. The absolute pointwise error between the FEM solution and PINN predicted solution is shown on the right side. Temperature predicted outside the expected bound (if any), i.e. when $u \notin [0, 1]$ is shown using the white and grey colour.

The FCNNs are continuous and differentiable functions and can't predict discontinuities such as the corners with conflicting BCs. Also, it is very challenging to obtain a trained PINN model with minimum validation error because we are not using the ground truth to calculate the training loss.

3.4.2 Model 2: DeepXDE's baseline PINN

We trained a FCNN with 8 hidden layers and 20 neurons in each layer using the Adam optimiser with hyperbolic tangent activation and exponential decay of the learning rate. We used DeepXDE's default sampling method, i.e., Sobol sequence to sample 800 boundary points and 2500 collocation points, i.e. only about half the collocation points compared to Model 1.

Although the training loss converged in 5k epochs, we continued to train the model until 30k epochs to demonstrate the benefits of exponential decay of the learning rate. As the training progresses, the amount of perturbation in the weights (as per equation 2.6) decreases. Thus, we don't see much difference in the predicted temperature distribution after 5k epochs, which was not possible with Model 1.

As discussed in Section 3.2.1, DeepXDE does not sample the corner points, so we ignored these points from the FEM solution to compute the relative L^2 error. Thus we obtain very low relative L^2 error as most of the error occurs around the corners. Figure 3.4(b) shows the predicted temperature distribution and the pointwise absolute error for the 2D steady-state heat conduction problem.

3.4.3 Models 3-11: NVIDIA Modulus

We used NVIDIA Modulus to train PINN frameworks from Model 3 to Model 11. We started with NVIDIA Modulus's baseline PINN with the integral form of the loss function (Model 3). Then we applied the SDF weights on interior points (Model 4) and on both interior and boundary points (Model 5). We used additional tools such as importance sampling (Model 6) and combined the importance sampling with adaptive activation and quasi-random sampling (Model 7). We also used the Fourier network (Model 8), SiReNs network (Model 9), Modified-Fourier network (Model 10) and DGM architectures (Model 11) with adaptive activation, importance sampling, quasi-random sampling and full SDF weights, i.e., SDF weights on both interior and boundary points with an exception for SiReNs network which has no adaptive activation implemented in NVIDIA Modulus 22.03. Furthermore, NVIDIA Modulus 22.03 only implements the Halton sequence to generate a quasi-random sample.

The SDF weights are manually adjusted depending on the problem and is an active field of research. We formulated the SDF weights on boundary points such that the weights on the corner points is zero and increased as we move away from the corner (see Equation 3.5). The SDF weights on the interior points depends on the shape of the spatial domain. We used the default SDF weights on collocation/interior points. Figure 3.5 shows the magnitude of the SDF weights

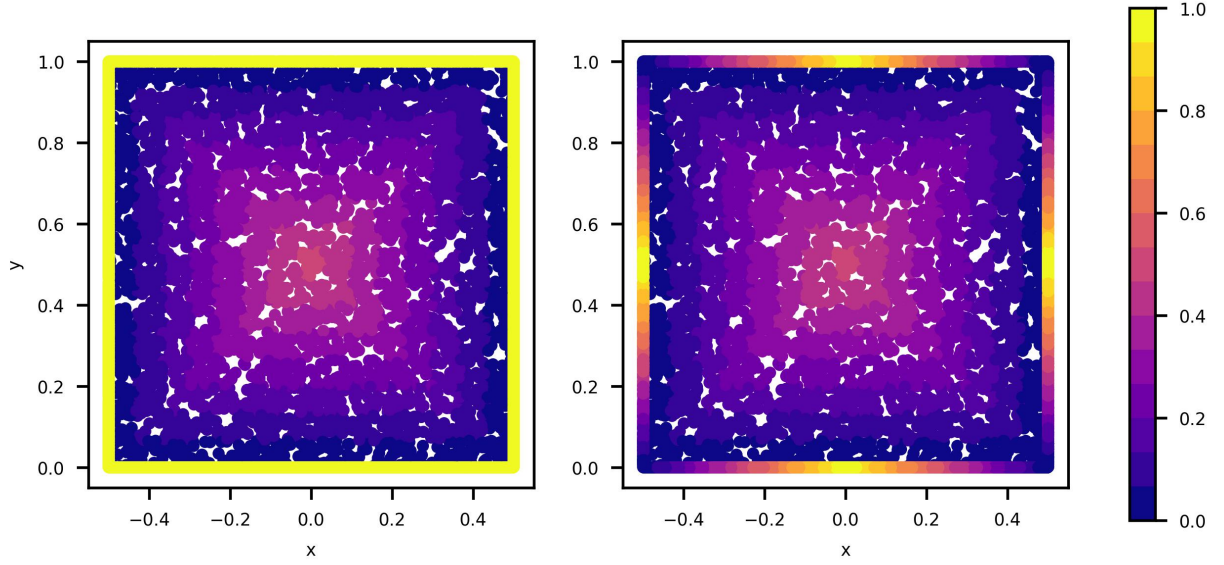


Figure 3.5: The magnitude of SDF weights on interior and boundary points of a square domain. The SDF weights for Model 4, i.e. with only interior points is shown on the left side. Whereas, the SDF weights with interior and boundary points is shown on the right side (for Model 5 to 11).

on the boundary and interior points of the square domain for Problem 1. In the case of a full set of SDF weights, it is worth noting that the maximum magnitude of the SDF weights on the interior points is 0.5. So, We are giving 50% less importance to the interior points compared to the boundary points, because we want the optimiser to obtain more information from the BC loss, as it does not converge in the case of Model 1 (as shown in Figure 3.3).

$$\text{Top and bottom boundary: } y = 1.0 - 2|x|$$

$$\text{Left and right boundary: } x = 1.0 - 2|y - 0.5| \quad (3.5)$$

In Model 3, Model 4 and Model 5, we trained NVIDIA Modulus's baseline PINN with default values, i.e. with 6 hidden layers and 512 neurons in each layer using the Adam optimiser with exponential decay in the learning rate and SiLU activation (Sigmoid Linear Unit) [145] for 20k epochs. We used 4000 boundary points and 4000 collocation points sampled from a uniform distribution. We used the same number of boundary points and collocation points in models from Model 3 to Model 11. We obtained a relative L^2 error of 0.08931, 0.10124, 0.09557 for Model 3, Model 4 and Model 5 respectively.

In Model 6, we replaced the uniform sampling with importance sampling to resample the

interior points in Model 5. We observed a significant improvement around the corners, resulting in a relative L^2 error of 0.06448. In the case of Model 7, we used adaptive activation with quasi-random sampling for the initial sampling in each batch and importance sampling for resampling of the interior points. We obtained a relative L^2 error of 0.06433, not a significant improvement over Model 6.

In Model 8, we trained a Fourier network with 10 Fourier features (see Section 2.8.1) using 6 hidden layers and 512 neurons in each layer using the Adam optimiser with SiLU adaptive activation for 20k epochs, with full SDF weights, quasi-random sampling and importance sampling. We used 10, 15, 25 and 35 Fourier features for the input encoding. We observed that the Fourier network is only working for 10 Fourier features and the absolute pointwise error is 0.18741, which is even higher than Model 1. For 15, 25 and 35 Fourier features the predicted temperature distribution were close to 0.5 over the entire domain, i.e. the average of upper and lower bound temperature in the domain.

In Model 9, we trained a SiReNs network (Section 2.8.3) with 6 hidden layers and 512 neurons in each layer using the Adam optimiser with Sine activation for 20k epochs, with full SDF weights, quasi-random sampling and importance sampling. The network was continuously experiencing exploding gradients until we reduced the learning rate to (2×10^{-5}) . Still after 12k epochs the training loss would abruptly increase, leading to prohibitively large absolute pointwise error. Hence, we forced the training to stop around 11k epochs and the relative L^2 error was found to be 0.12640.

In Model 10, we trained a modified Fourier network (Section 2.8.2) with 6 hidden layers and 512 neurons in each layer using the Adam optimiser with SINE activation for 20k epochs, with full SDF weights, quasi-random sampling and importance sampling. Similar to Model 10, we used 10 Fourier features for the input encoding. The predicted temperature distribution looks similar to Model 8 and the relative L^2 error was found to be 0.17577.

In Model 11, we trained a DGM architecture (see Section 2.8.3.1), with 6 hidden layers and 512 neurons in each layer using the Adam optimiser with Sine activation for 20k epochs, with full SDF weights, quasi-random sampling and importance sampling. We obtained a relative L^2 error of 0.08229.

In Figure 3.4, Model 6, 7 and 11 resulted in less than 1% absolute pointwise error at most of the points in the domain. This indicates that the SDF weights and the importance sampling plays an important role in solving stiff-PDEs with a discontinuous solution. In Table 3.2, Model 3, 5, 6, 7 and 11 resulted in less than 10% relative L^2 error. Further investigation is required to determine whether the DGM architecture is advantageous compared to the baseline PINNs in

higher dimensions.

3.4.4 Hard constrained BCs

So far, we trained Problem 1 with soft constrained BCs, i.e. the BC loss is added as a term in the loss function. In this section, we discuss the exact imposition of BCs in PINNs. Hard constrained BCs involves the construction of a continuous and differentiable function through which we pass the output of the NN. Problem 1, involves discontinuous BCs which can't be exactly satisfied with a continuous and differentiable function. However, it is possible to satisfy the BCs on two opposite walls, here we chose to exactly satisfy the top and bottom walls using the following output transform function.

$$\hat{u} := y(y - 1) \hat{u} \quad (3.6)$$

We trained the Model 2 (see Table 3.1) with 3500 collocation points and 400 boundary points with Adam for 30k epochs. Figure 3.6 shows the DeepXDE predicted solution to Problem 1 with the hard constrained BCs at the top and bottom walls. Due to hard constrained BCs at the top and bottom walls, we observe significant errors around the left and right walls resulting in a relative L^2 error of 0.183119. Thus, it is not recommended to apply hard constraints to the BCs in stiff-PDEs.

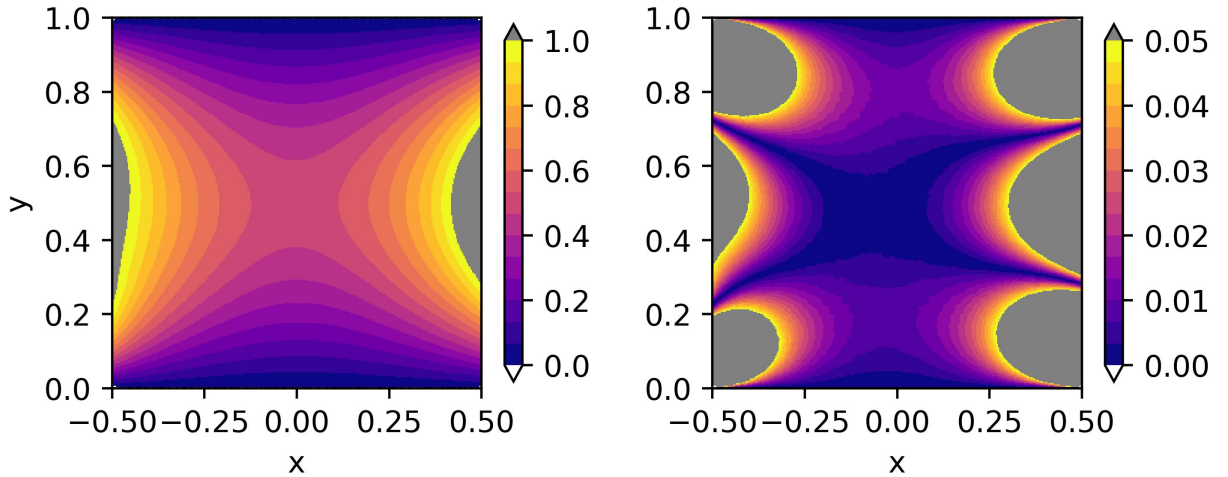


Figure 3.6: DeepXDE predicted solution of 2D steady-state heat conduction problem (Equation 3.4) with hard constrained BCs at the top and bottom walls on the left and the absolute pointwise error between the FEM and PINN predicted solutions is shown on the right side. Temperature predicted outside the expected bound (if any), i.e. when $u \notin [0, 1]$ is shown using the white and grey colour.

3.4.5 Overfitted solution

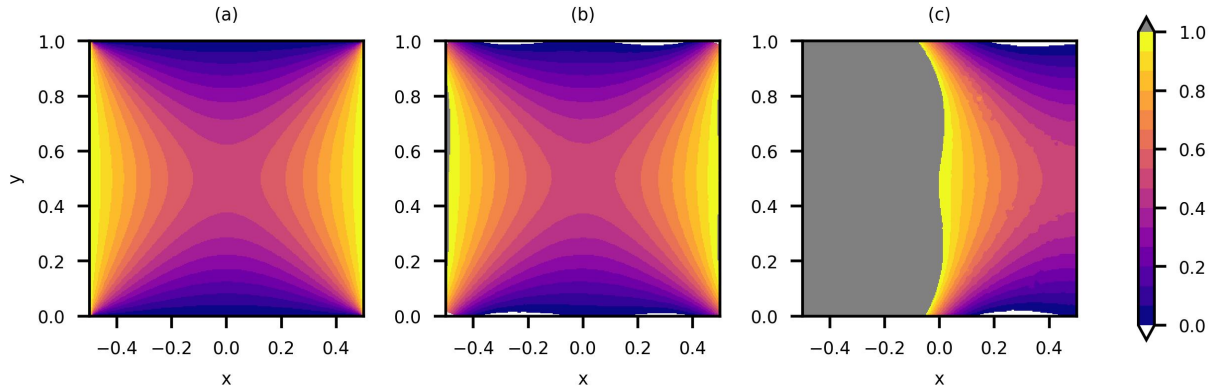


Figure 3.7: Comparison of PINN predicted solution from the training dataset and at new data-points. (a) The FEM solution of Problem 1 (Equation 3.4), (b) The DeepXDE predicted solution of the same problem from the training dataset, and (c) the DeepXDE inferred solution at new locations in the domain using the pretrained PINN from (b). Temperature predicted outside the expected bound (if any), i.e. when $u \notin [0, 1]$ is shown using the white and grey colour.

In this section, we discuss the behaviour of an overfitted PINN model while predicting the solution of the PDE at new spatio-temporal locations in the domain. We trained the Model 2 (see Table 3.1) with 2500 collocation points and 200 boundary points with Adam for 30k epochs. Figure 3.7 shows the DeepXDE predicted solution to Problem 1 from both the training dataset and new locations in the domain within the domain. Given that the trained model's accuracy during the validation is low, we will categorise the model as an overfitted model. The validation loss can be decreased by adding more points in the training data set, for instance, NVIDIA Modulus samples different points for the training in each epoch. Intuitively, the overfitted model is computationally cheaper than a generalised model as it requires very few collocation points, which is evident with this example.

3.5 Problem 2: 3D steady-state heat conduction

The next problem we address is a 3D steady-state heat conduction problem with conflicting BCs at the edges and the corners. We solved the problem with different PINN frameworks and compared the results to the FEM solution. The 3D heat conduction problem is described as follows:

$$\begin{aligned}
u_{xx} + u_{yy} + u_{zz} &= 0, & x \in [-0.5, 0.5], & y \in [-0.5, 0.5], & z \in [0, 1] \\
u(x, y, 0) &= u(x, y, 1) = u(0.5, y, z) = 0 \\
u(-0.5, y, z) &= u(x, -0.5, z) = u(x, 0.5, z) = 1
\end{aligned} \tag{3.7}$$

We used MATLAB Partial Differential Equation Toolbox [143] to solve the 3D steady-state heat conduction problem (Equation 3.7) with quadratic triangles (Figure 3.8).

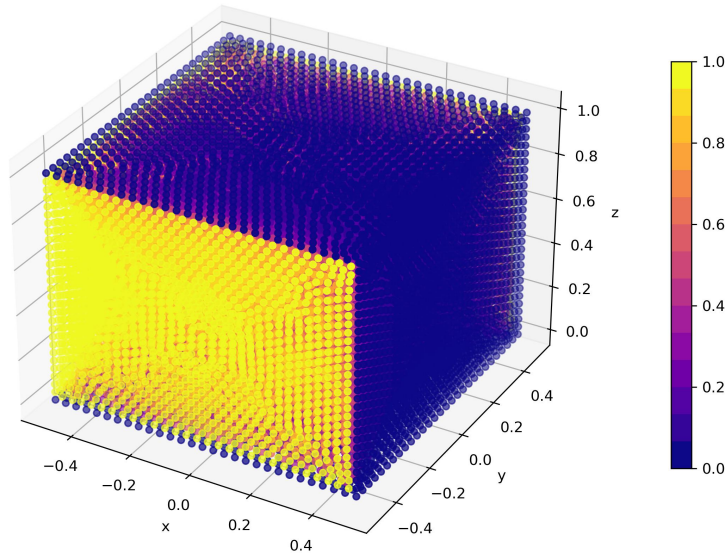


Figure 3.8: FEM solution for the 3D steady-state heat conduction problem (Equation 3.7) using MATLAB Partial Differential Equation Toolbox.

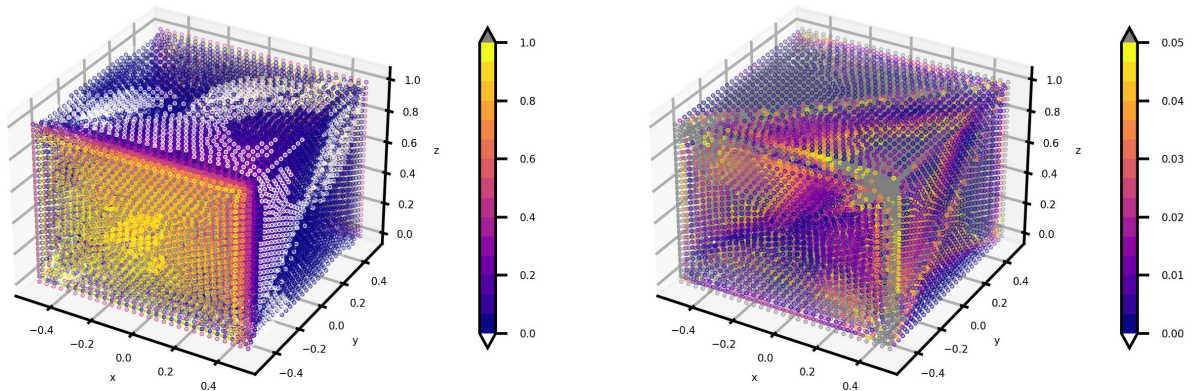
We again used the same PINN models (see Table 3.1) to solve the 3D steady-state heat conduction problem. We did not use the full SDF weights because the boundary walls are planes instead of lines in 2D. This is where we potentially require an algorithm, instead of manually constructing the SDF weights for the boundary points. Therefore, for this problem, we refer to the SDF weights on interior points as the full SDF weights and we exclude Model 4. Table 3.3 summarises the network parameters and relative L^2 error for various PINN frameworks. Figure 3.9 shows the solution of the 3D steady-state heat conduction problem for different PINN frameworks.

Table 3.3: Problem 2: Summary of training parameters and relative L^2 error for different PINN frameworks

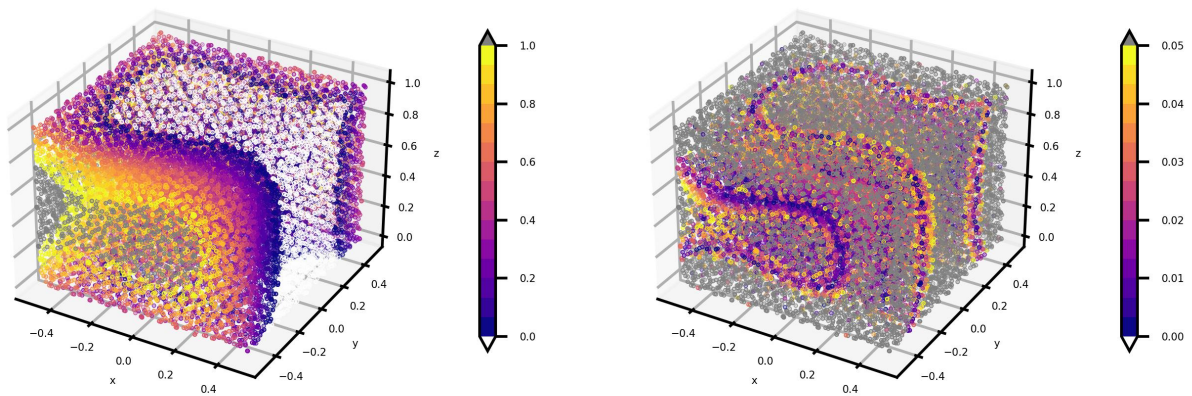
	Layers	Nodes	Boundary points	Collocation points	α	Epochs	Relative L^2 error
Model 1	8	150	5814	34071 *	1e-3	20k	0.23778
Model 2	10	150	5000	15000**	1e-3	15k	3.15547
Model 3	6	512	4000	4000	1e-3	20k	0.12031
Model 5	6	512	4000	4000	1e-3	10k	0.07455
Model 6	6	512	4000	4000	1e-3	20k	0.07163
Model 7	6	512	4000	4000	1e-3	20k	0.08557
Model 8	6	512	4000	4000	1e-3	20k	0.07302
Model 9	6	512	4000	4000	2e-5	20k	0.56448
Model 10	6	512	4000	4000	1e-3	20k	0.09498
Model 11	6	512	4000	4000	1e-3	30k	0.07213

* Model 1 uses the nodal coordinates from the FEM mesh.

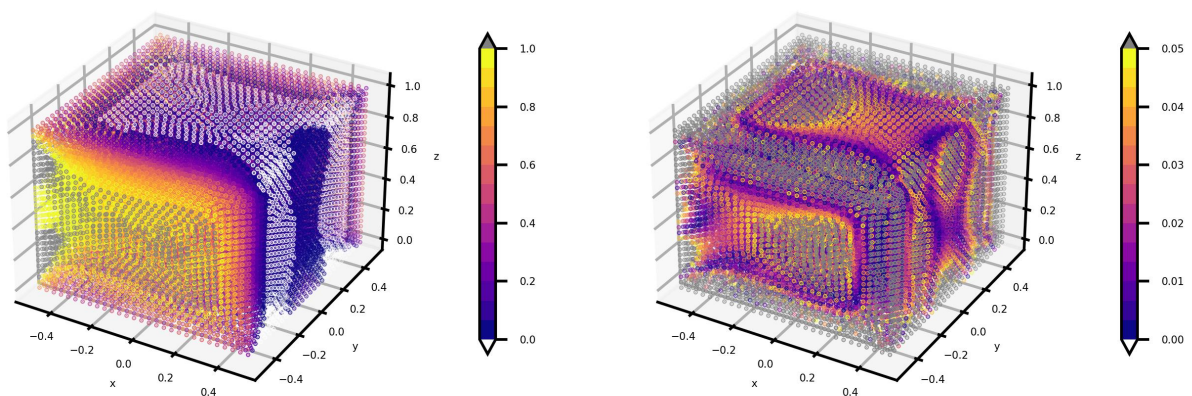
** Model 2 uses the Sobol sequence.



(a) Model 1

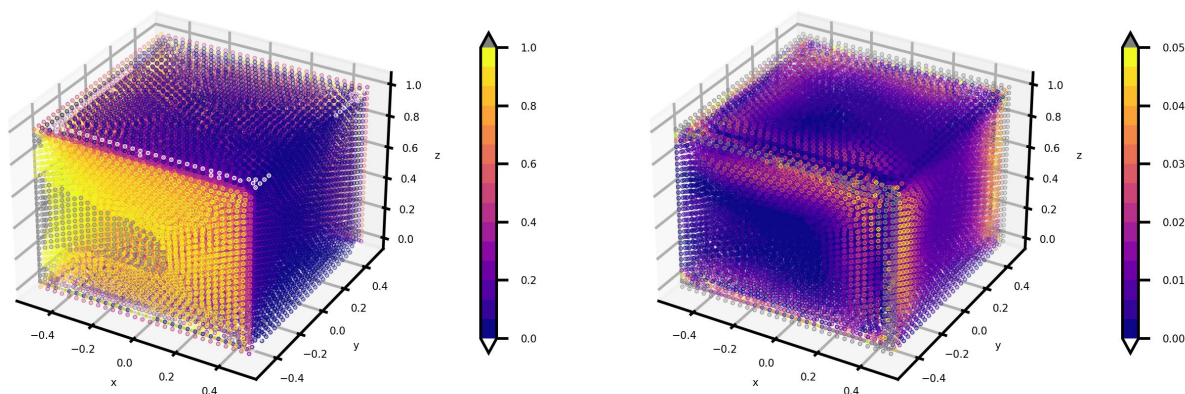


(b) Model 2

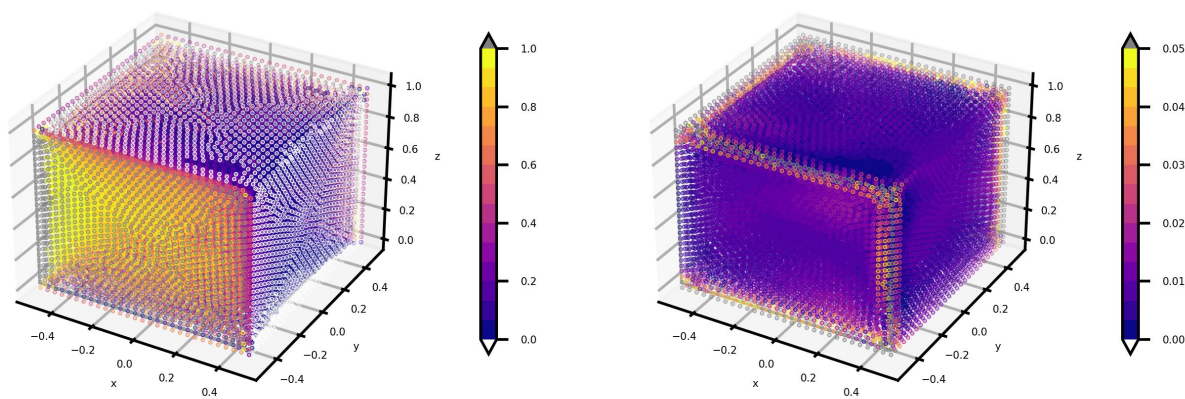


(c) Model 3

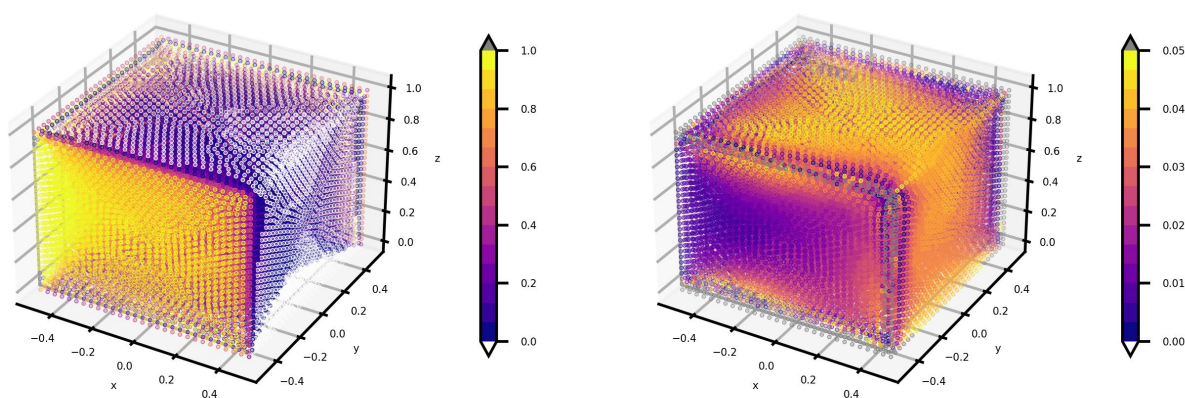
Figure 3.9: Continued on next page.



(d) Model 5

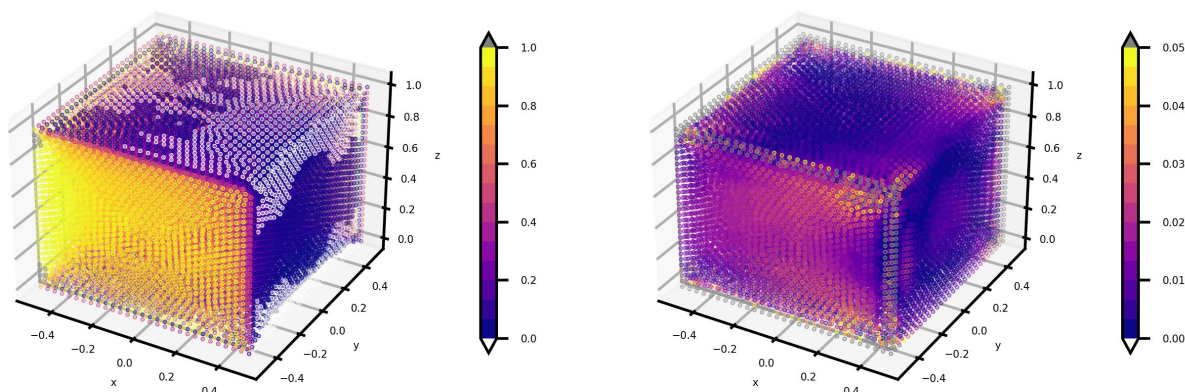


(e) Model 6

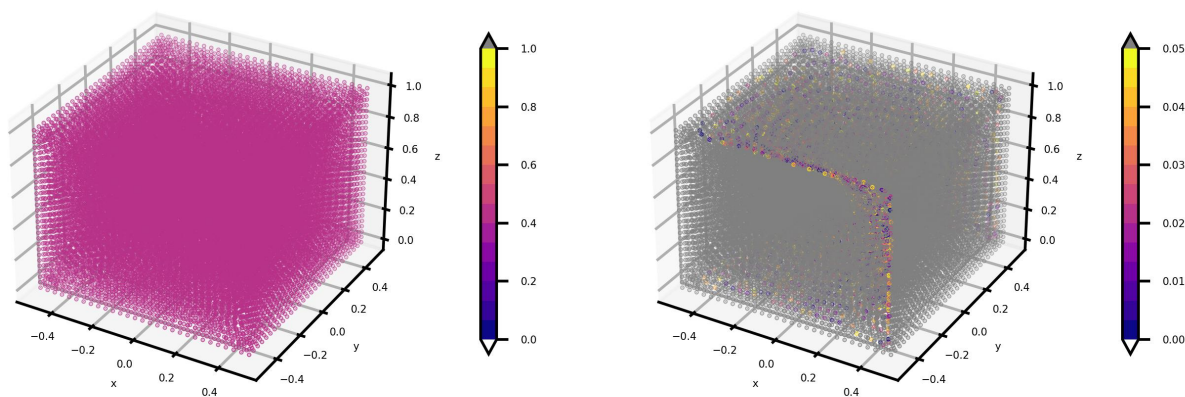


(f) Model 7

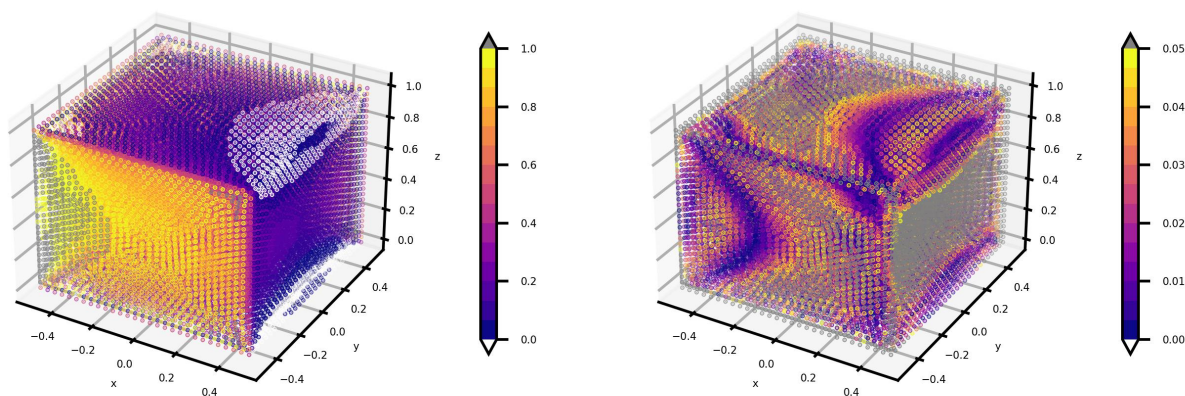
Figure 3.9: Continued on next page.



(g) Model 8



(h) Model 9



(i) Model 10

Figure 3.9: Continued on next page.

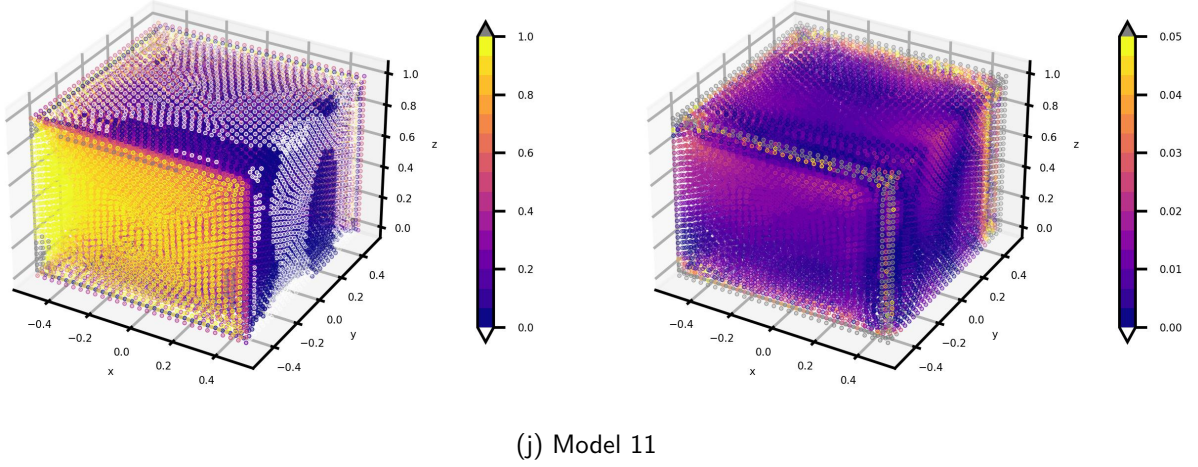


Figure 3.9: PINN predicted solution of 3D steady-state heat conduction problem is shown on the left side. The absolute pointwise error between the FEM solution and PINN predicted solution is shown on the right side. Temperature predicted outside the expected bound (if any), i.e. when $u \notin [0, 1]$ is shown using the white and grey colour.

In Problem 1 (Section 3.4), the discontinuities occurred at the corners of the square domain, which affected only a few training points. Whereas, in Problem 2, the discontinuities affected not only the vertices but also the edges. We can sample a large number of points along these edges. Thus, Problem 2 is more suitable for testing the robustness of different PINN frameworks.

In Problem 2, all the models had the same number of layers, nodes per layer and the learning rate as in Problem 1. However, we increased the number of boundary points and collocation/interior points in Model 1 and 2.

In Model 1, we observed that the interior points had significant absolute pointwise error, meaning the PDE loss didn't converge. We obtained a relative L^2 error of 0.23778. In Model 2, the DeepXDE network didn't sample the points at the edges and vertices (see Section 3.2.1). We obtained a relative L^2 error of 3.15547.

In the case of models trained within NVIDIA Modulus, the SDF weights on the interior nodes from Problem 1 was extended to three dimension such that interior points close to the boundary were assigned negligible weights. As we move away from the boundary, the SDF weights on the interior points increases until it reaches close to 0.5 around the centroid of the cubical domain (see Figure 3.10).

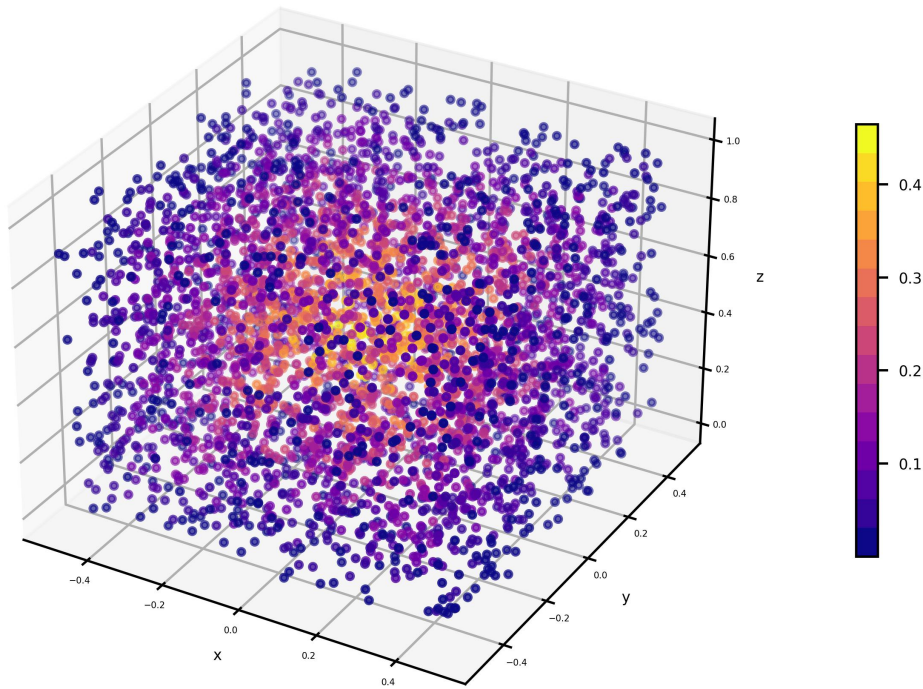


Figure 3.10: The magnitude of SDF weights on interior points of the cubical domain of problem 2.

NVIDIA Modulus's baseline PINN (Model 3) predicts better temperature distribution compared to Models 1 and 2 with a relative L^2 error of 0.12031. The addition of SDF weights on the interior points (Model 5) dramatically reduces the relative L^2 error to 0.07455.

To obtain accurate results, the nodal coordinates or the training data needs special treatment either by transforming the coordinates to higher dimensions using the Fourier network or by adding more transformer layers using the DGM architecture or both using the modified Fourier network. From Table 3.3, it is clear that the baseline PINN without SDF weights is not suitable for solving stiff-PDEs with discontinuities in 3D.

In summary, Model 5, 6, 7, 8, 10 and 11 resulted in less than 10% relative L^2 error and less than 5% absolute pointwise error at most of the points in the domain. It is worth mentioning that the SiReNs network (Model 9), which uses sine activation functions, struggled to minimise the loss function effectively. This difficulty arises from the spectral bias of the sine activation, which tends to prioritise low-frequency components of the solution. As a result, the network was unable to accurately capture the sharp transitions required by the problem and instead predicted a nearly constant temperature. In contrast, the tanh-based models handled the discontinuities more

effectively due to the monotonic nature of the tanh activation and its tendency to saturate at sharp transitions. Consequently, Model 9 resulted in a relative L^2 error of 0.56448 (see Figure 3.9).

A summary of the total training time (sec) and training time (sec) per epoch in Problem 1 and 2 for each model is presented in Table 3.4. The training time does not include the time taken in pre-processing of the data and initialisation of the neural network. In Model 1 and 2, the number of data points were different in Problem 1 and 2, which influences the training time. Thus, we could not draw a concrete conclusion. In Model 3-10, even though the number of data points were same there was no noticeable increase in the training time per epoch. However, in Model 11, the training time per epoch was almost 3 times for Problem 2, a 3D problem, compared to Problem 1, a 2D problem. This is because one DGM layer contains 8 weight matrices and introducing one more feature into the training dataset increases the number of operations exponentially.

Table 3.4: Summary of the total training time (sec) and training time (sec) per epoch in Problem 1 and 2 for each model.

	Problem 1		Problem 2	
	Training time (sec)	Training time/epoch (10^{-3} sec)	Training time (sec)	Training time/epoch (10^{-3} sec)
Model 1	455	15.61	2259	112.95
Model 2	482	16.06	845	56.33
Model 3	632	31.60	592	29.6
Model 4	527	52.71	-	-
Model 5	530	53.00	585	58.5
Model 6	2759	137.95	1931	96.55
Model 7	1843	92.15	2193	109.65
Model 8	2031	101.55	2416	120.80
Model 9	769	69.91	1144	57.2
Model 10	2792	139.60	3481	174.05
Model 11	2670	89.00	8174	272.46

3.6 Parametric heat conduction problem

As discussed in Section 2.7.1, PINNs can be parameterised by adding the parameter of interest as another feature in the training dataset. We solved two parametric 2D steady-state heat conduction problems with parameterised conductivity and parameterised geometry (see Table 3.5).

Table 3.5: Summary of parametric heat conduction problems

	Parametric conductivity	Parametric geometry
PINN framework	Model 2	Model 7, Model 10, Model 11
Layers	8	6
Nodes per layer	20	512
Boundary points	10000	4000
Collocation points	25000	4000
α	1e-3	1e-3
Epochs	30k	20k
Average relative L^2 error	0.10305	0.05460, 0.10407, 0.08192

3.6.1 Problem 3: Parameterised conductivity

We formulated the 2D steady-state heat conduction problem such that the conductivity κ is varying from 0 to 1 in Equation 3.8.

$$\begin{aligned}
 u_{xx} + \kappa u_{yy} &= 0, & x \in [-0.5, 0.5], & y \in [0, 1], & \kappa \in [0, 1] \\
 u(x, 0) = u(x, 1) &= 0 & u(-0.5, y) = u(0.5, y) &= 1
 \end{aligned} \tag{3.8}$$

We trained problem 3, with various models from Table 3.1 and observed that the predicted temperature distribution is fairly accurate except for model 1 because we did not use a learning rate scheduler (see Section 3.4.1). The absolute pointwise error is less than 5% on the interior points with some errors around the corners, similar to Problems 1 and 2. Figure 3.11 shows the solution to Problem 3 using Model 2. Model 2 predicted a reasonably accurate temperature distribution in just 523 seconds. Model 3 to 11 took 2-3 times the time to train the model with very little improvement.

3.6.2 Problem 4: Parameterised geometry

Similar to conductivity the geometry can also be parameterised. We used the y-dimension of the 2D steady-state heat conduction problem as the parameter (Equation 3.9).

$$\begin{aligned}
 u_{xx} + u_{yy} &= 0, & x \in [-0.5, 0.5], & y \in [0, L], & L \in [1, 10] \\
 u(x, 0) = u(x, L) &= 0 & u(-0.5, y) = u(0.5, y) &= 1
 \end{aligned} \tag{3.9}$$

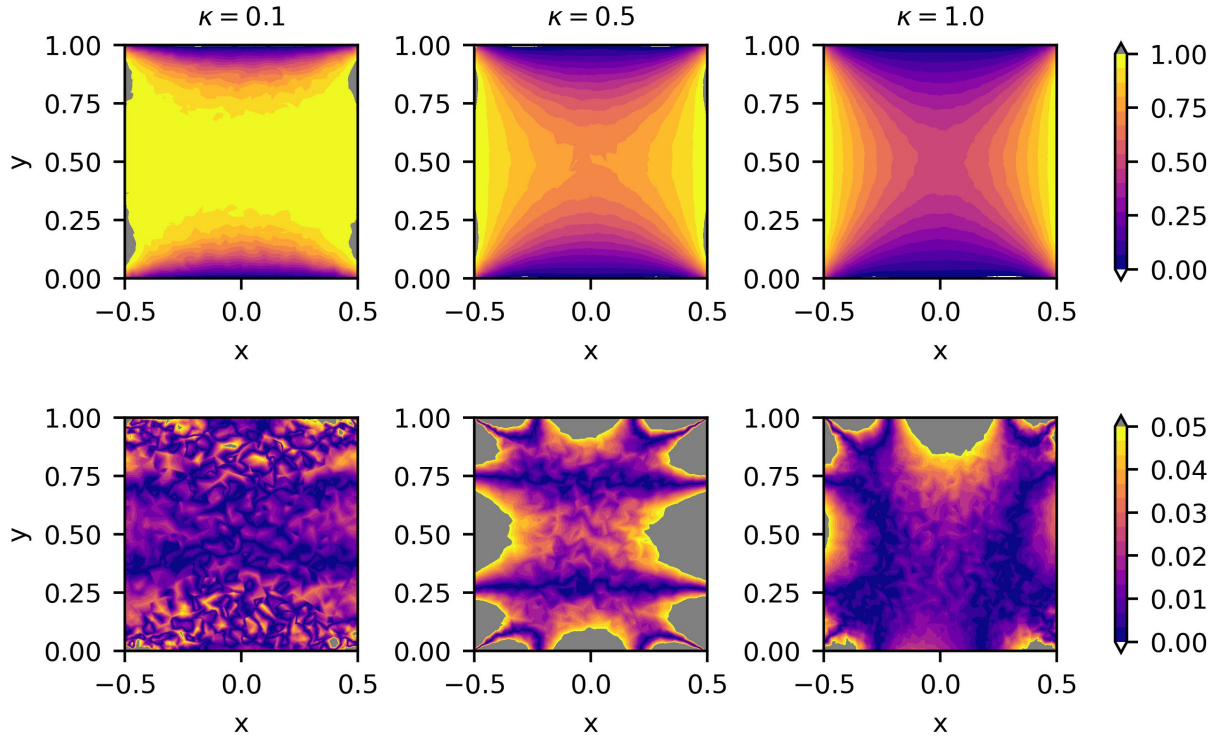


Figure 3.11: Solution to Problem 3 using Model 2. The predicted temperature distribution (on the top) and the absolute pointwise error (on the bottom) for different values of κ (see Equation 3.8). Temperature predicted outside the expected bound, i.e. when $u \notin [0, 1]$, is shown in white and grey colour.

Implementing the parametric geometry is not straightforward, care should be taken that for each geometry parameter the sampled points along the geometry parameter is within the spatio-temporal domain. For example, as per Equation 3.9, $y \in [0, L]$ and $L \in [1, 10]$, i.e. $y \leq L$ for each sample in the training dataset.

Figure 3.12 shows the boundary points sampling of Problem 4 to illustrate the implementation of parametric geometry. Here, x and y axes shows the 2D spatial domain for each geometry parameter along the L axis. As discussed earlier, for $y \leq L$, this results in moving the upper boundary on the y axis as L increases.

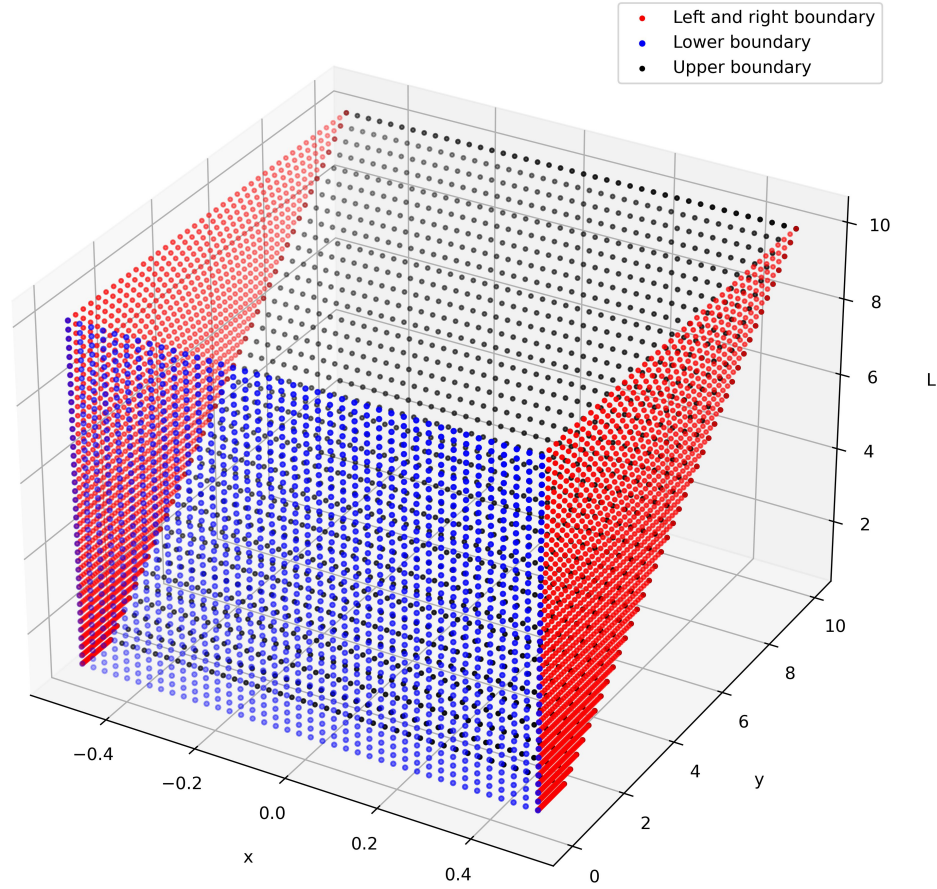
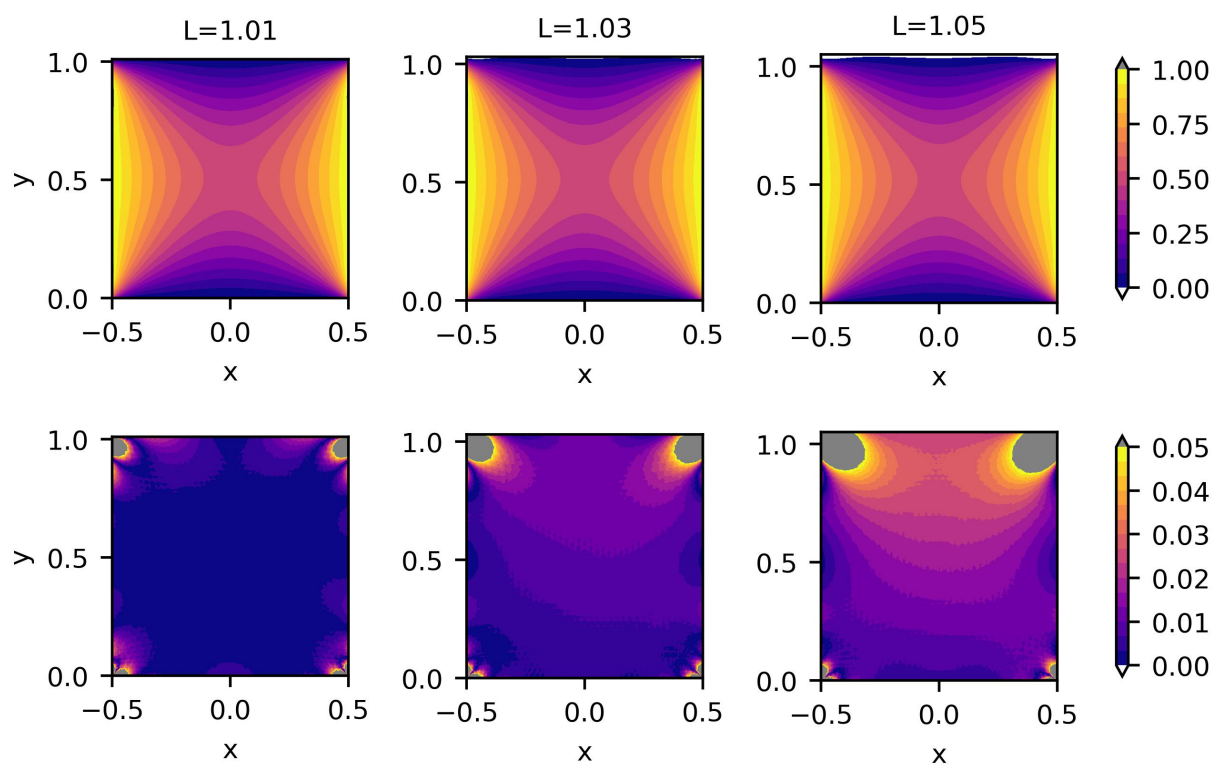


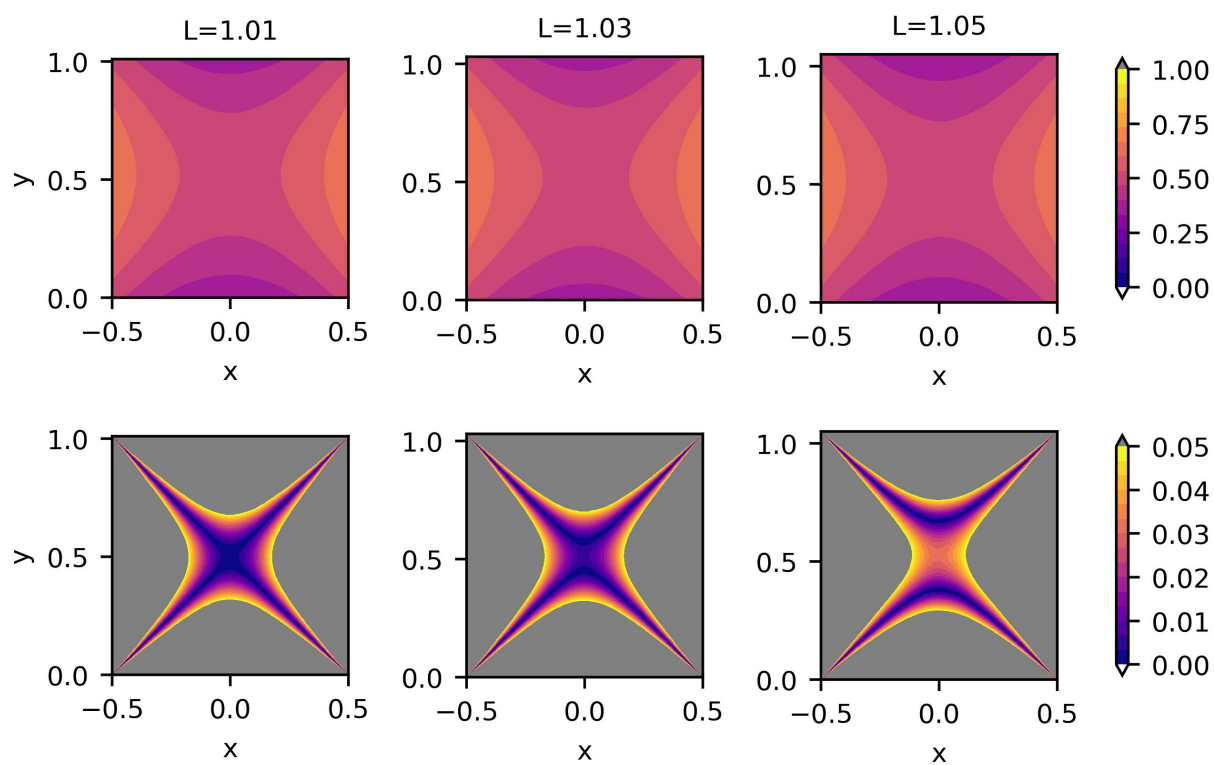
Figure 3.12: The boundary points sampling of the parameterised geometry in Problem 4.

We observed that none of the models could predict a temperature distribution that visually looks similar to the FEM solution. Some of the models such as Model 1, 3, 4, 5, 6 and 7 could not converge the loss. Whereas, model 10 and 11 resulted in gradient explosion. We think that none of the PINN frameworks in Table 3.1 does have enough features and parameters to predict multiple discontinuous temperature distribution around the moving upper boundary. We need more sophisticated PINN frameworks to solve these types of problems. Thus, we decreased the range of the parameter L until we could predict a temperature distribution that visually resembles the FEM solution. The parameter's range was eventually narrowed to 1.05, i.e. $L \in [1, 1.05]$.

We trained Model 7, 10 and 11 with the redefined L parameter. For Model 10 and Model 11, we have also shown the predicted temperature distribution without the SDF weights to highlight how important they are while solving problems with discontinuities. Figure 3.13 shows the predicted temperature distribution for the redefined problem with Model 7, 10 and 11 with and without the SDF weights.

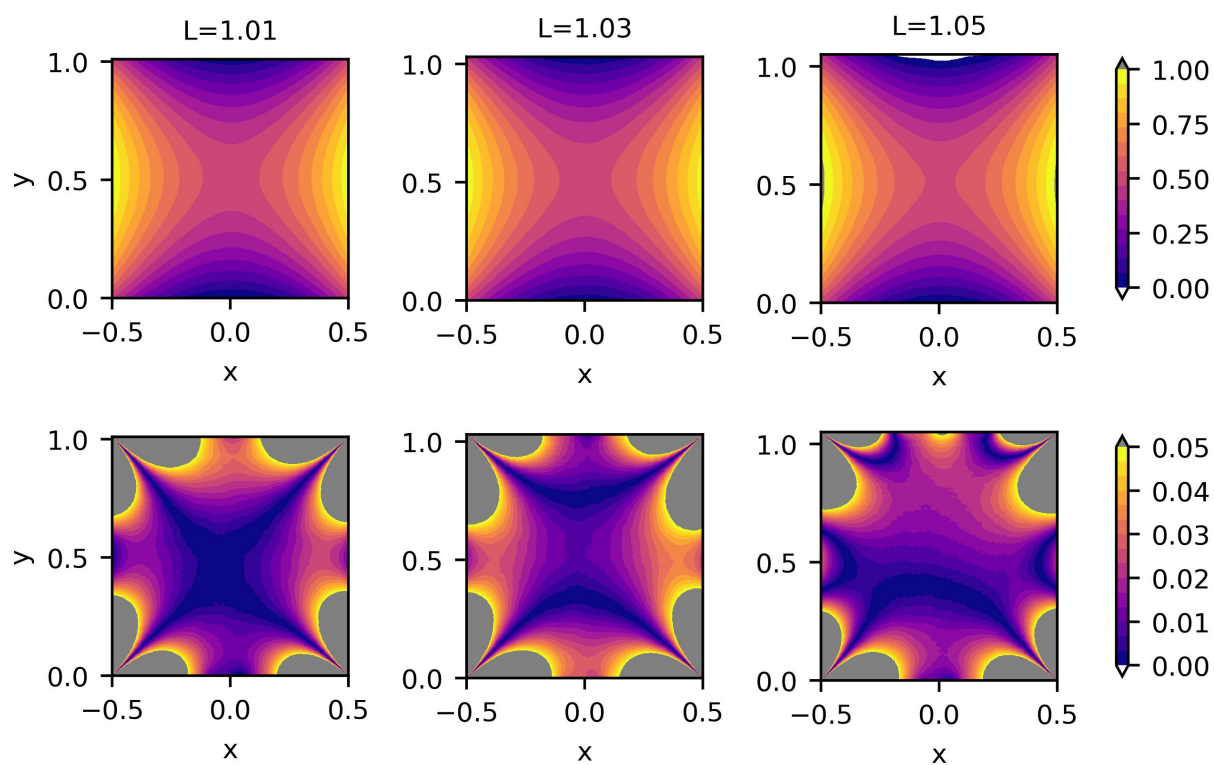


(a) Model 7

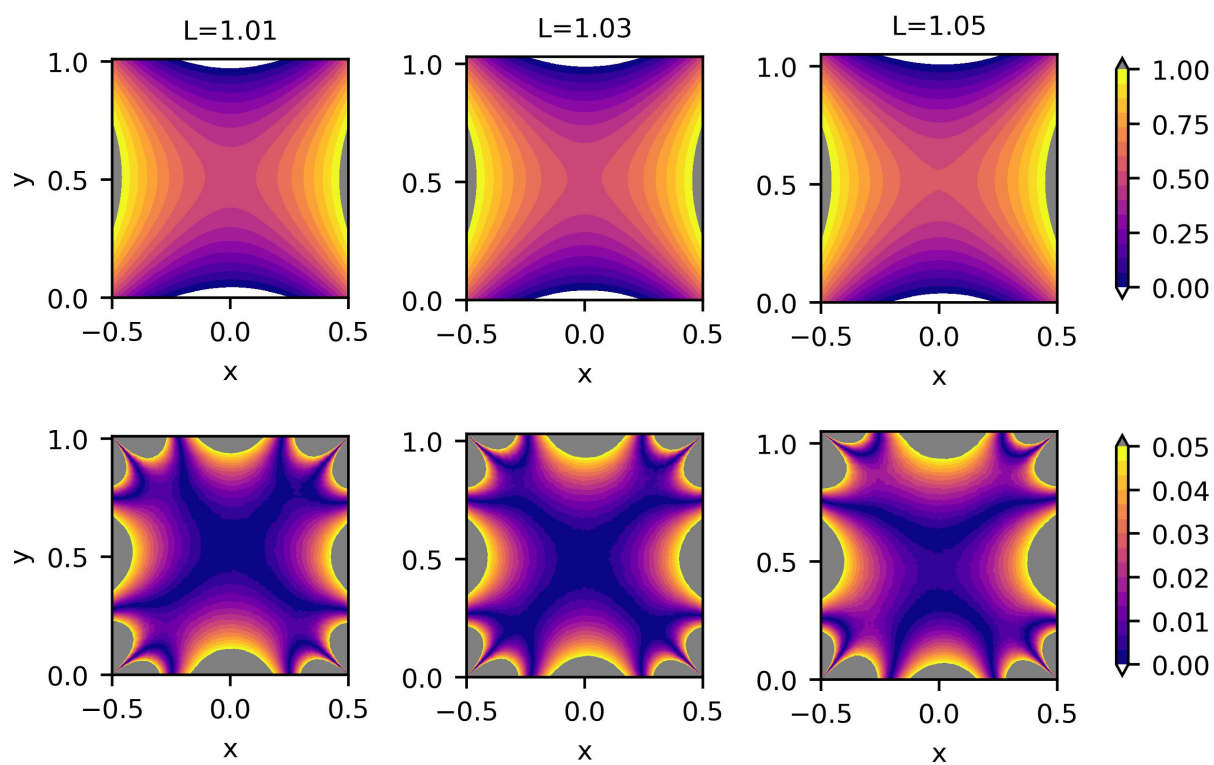


(b) Model 10 without SDF weights

Figure 3.13: Continued on next page.



(c) Model 10



(d) Model 11 without SDF weights

Figure 3.13: Continued on next page.

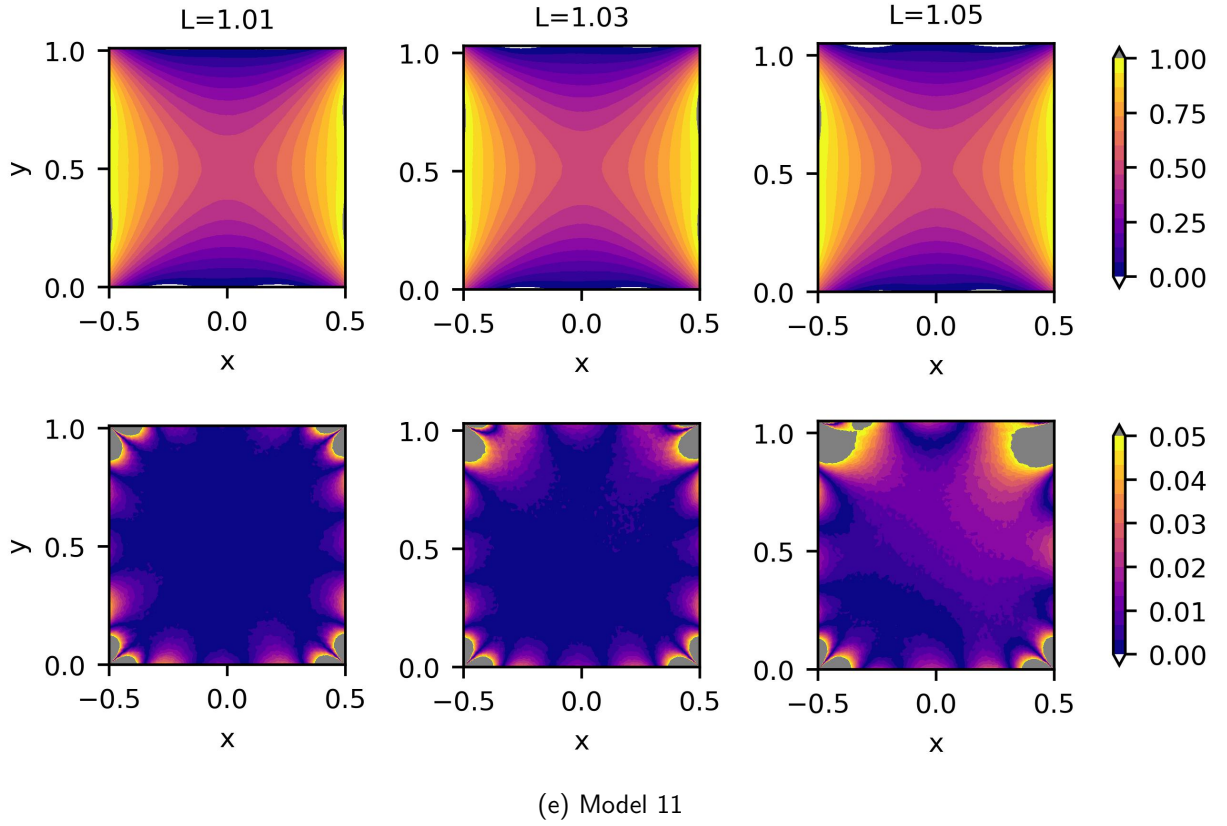


Figure 3.13: Solution of 2D steady-state heat conduction problem with parameterised upper boundary. Each plot shows the predicted temperature distribution for a specific value of parameter $L \in [1, 1.05]$. Temperature predicted outside the expected bound, i.e. when $u \notin [0, 1]$, is shown using white and grey colour.

In Model 7, the predicted temperature distribution around the left and right boundary does not change with the moving upper boundary, especially, around the upper-left and upper-right corners of the square domain. In Model 10 without the SDF weights, the predicted temperature distribution does not visually resemble the FEM solution. Whereas, when we use the SDF weights, the predicted temperature distribution improves dramatically.

In Model 11, the predicted temperature distribution without the SDF weights was much better than any other Model without the SDF weights. Also, Model 11 with SDF weights predicts better temperature distribution compared to Model 7 and 10 with SDF weights.

3.7 Conclusion

In this thesis, we have reviewed application of PINNs to stiff-PDEs, specifically for simple steady-state heat conduction problems with discontinuous BCs at the corners. We defined a list of PINN

frameworks in Table 3.1 which included the baseline PINN and models from open-source libraries such as DeepXDE and NVIDIA Modulus.

We started with a 2D steady-state heat conduction problem (Problem 1). We observed that both the baseline PINN and DGM architecture predicted temperature distribution with 5-10% absolute pointwise error in the corner regions for most models. The results are summarised in Table 3.2.

Next, we solved Problem 1 with hard constrained BCs. We showed that we can't satisfy all the BCs exactly when they conflict with each other. Problem 1 contains discontinuous BCs, which can't be satisfied exactly with a differentiable function such as a Neural Network. Thus, we do not recommend applying hard constrained BCs on a stiff-PDE.

We also demonstrated the inability of an overfitted PINN to predict the temperature distribution at new locations in the domain from a pretrained PINN. However, an overfitted PINN proves to be computationally cheap when the solution is to be inferred on the training dataset only.

We then solved a 3D steady-state heat conduction problem (Problem 2), an equivalent of Problem 1 in 3D space. This is where modified PINN frameworks such as Fourier network, Modified Fourier network, DGM architecture proves to be more accurate than the baseline PINN frameworks (Model 1-7) in terms of the relative L^2 error. Furthermore, the SDF weights significantly improved the accuracy. This is primarily because the modified PINN frameworks are inherently evolutionary over the baseline PINN. The Fourier network addresses the spectral bias in the baseline PINNs, the DGM architecture is useful for solving problems in higher dimensions and the Modified Fourier network is a combination of both.

Then we solved Problem 1 with parametric conductivity (Problem 3) using DeepXDE's baseline PINN. DeepXDE's baseline PINN accurately predicted the temperature distribution while using fewer resources compared to Model 3-11.

We also solved Problem 1 with parametric geometry by extending the y-dimension (Problem 4). We observed that solving stiff-PDEs with parametric geometry can be very challenging for PINNs. Even robust PINN frameworks such as the Modified Fourier Network resulted in exploding gradients. In Problem 4, we also showed that the predicted temperature distribution with SDF weights significantly outperforms those without the SDF weights. Table 3.5 summarises the network parameters and the accuracy (relative L^2 error) of various models for Problem 3 and 4.

Among the PINN frameworks (see Section 3.1), baseline PINNs are not useful in the case of 3D problems. On the other hand, PINN frameworks such as Modified Fourier network and DGM architecture are robust even on 3D problems. The SiReNs network is very unstable and often results in exploding gradients even with exponential decay of the learning rate. The SiReNs

network fails to predict the temperature distribution in Problem 2 even with a reduced learning rate. A summary of the best models for each problem is shown in Table 3.6.

Table 3.6: Summary of models with least relative L^2 error.

	Models
Problem 1	3, 5, 6, 7 and 11
Problem 2	5, 6, 7, 8, 10 and 11
Problem 3	All except Model 1
Problem 4	7, 10 and 11

In Table 3.6, Model 7 and 11 can be categorised as the best performing models in general for the four problems we considered. Model 10 also works in most of the cases. Model 7 is computationally cheaper than Model 11 (see Table 3.4). However, Model 11 seems to work without SDF weights in Problem 4 which is an advantage for solving more complicated problems.

Among the additional tools (see Section 3.2), the SDF weights are an important asset for solving problems with discontinuous BCs. The importance sampling is a loss dependent resampling strategy of the collocation points. It samples points proportional to the absolute pointwise error, which is particularly useful in Stiff-PDEs where we need more points to capture the sharp gradients. The adaptive activation and quasi-random sampling improves the convergence and the accuracy respectively.

We also tried to solve 2D steady-state heat conduction problem with a temperature dependent conductivity along the y-direction. However, we could not converge the training loss with any of the models in Table 3.1. This difficulty can be attributed to the dynamic nature of the loss landscape in this scenario. As the temperature changes during training, the conductivity and thus the input to the network also changes, leading to a constantly shifting loss landscape. This made it challenging for the optimiser to find a stable minimum, resulting in non-convergence of the models.

The author of this thesis would like to acknowledge that while there was interest in solving Problem 1 with parametric BCs, it was initially deemed infeasible due to the lack of existing theoretical techniques or prior work. However, subsequent advancements in understanding have shown that the concept of pseudofeatures [146] could be used to solve such problems. Despite these developments, the implementation of such an approach was beyond the scope of the thesis and was therefore not pursued.

The development of FEM took us almost 5 decades. In contrast, PINNs are evolving rather quickly. As of now, PINNs are able to solve simple discontinuous problems in 2D and 3D without any problem-specific tuning and transfer learning. The same framework can be used to solve the

parametric PDEs which is difficult to achieve with conventional PDE solvers. We believe that in 2-3 years, we will see PINNs solving complex benchmark problems. There is a need to automate the signed distance function for problems with discontinuous solutions. Also, there is a lack of benchmarking in PINNs. We can not use the same parameters for all the Models in Table 3.1. To compare different PINN frameworks, an improvement to this field would be for the research community to collectively agree upon a standard set of benchmarks.

Chapter 4

Hyperparameter tuning of PINNs

4.1 Introduction

Despite its success in problems with smooth solutions [71, 84, 131], PINN and their variations [27, 74, 99, 147] still encounter fundamental issues. For instance, PINN is still lacking a guaranteed convergence criteria [148], and the correlation between the network hyperparameters and performance is elusive [149]. Moreover, PINNs encounter significant issues solving stiff-PDEs, especially, problems with discontinuous solutions [71, 150] as discussed in Chapter 3. Hence, there exists a necessity for studies that specifically address convergence of PINNs for problems with discontinuous solutions.

Generally, PINNs will not converge unless we mitigate the effect of the discontinuities. A technique to accomplish this involves the use of a signed distance function (SDF) and requires a prior understanding of the region of the discontinuity [119]. The SDF ensures that the points sampled close to the region of discontinuity are assigned with minuscule weights, effectively reducing the contribution of these regions towards the training loss.

Similar to any NN, PINNs have a number of hyperparameters such as, the learning rate, the activation function, the optimiser, the weights in the loss function, the width and depth of the NN. The high-dimensional hyperparameter search space makes it difficult to obtain optimised hyperparameters that are suitable for a broad range of problems. Therefore, it is necessary to select fewer hyperparameters and focus on a specific set of problems.

In some early attempts, researchers treated the coefficients of loss terms as hyperparameters to balance the contribution of each loss term [71, 82–84, 151]. The approach was computationally efficient as the hyperparameter optimisation did not involve a complicated algorithm instead it was updated through the same optimiser used to train the PINN. However, this approach had

limitations regarding the range of hyperparameters that could be chosen. It did not allow choosing hyperparameters such as the number of hidden layers or the neural network architecture.

Some researchers used grid search [152, 153] and Bayesian optimisation [154–156] to gain greater flexibility in selecting hyperparameters. However, with the increase of number of hyperparameters, the approach became computationally expensive due to the high-dimensional search space. In this study, we have chosen to employ manual tuning. It offers several advantages, including, full control over the hyperparameter values, better interpretability and computational efficiency in comparison to grid search or Bayesian optimisation methods, when only a few hyperparameters are studied.

Several studies conducted manual tuning with a wide range of hyperparameters [157, 158]. The outcomes were transferable to similar problems with minor variations, making them ideal starting points for problems in the same domain of physics. The present work aims to identify optimal hyperparameters for PINNs in order to improve their accuracy and convergence when solving heat transfer problems with discontinuous BCs. We performed hyperparameter optimisation to identify all possible settings of neural network architectures, number of hidden layers, activation functions and learning rates that could achieve a relative L2 error of 10% or less across all test cases. These optimal hyperparameters enable more accurate and efficient solutions to similar problems.

4.2 Summary of methodology

In machine learning, hyperparameters are parameters that influence the overall performance of the model. Hyperparameter optimisation is a process of selecting the optimal combination of hyperparameters that minimise or maximise an objective function such as the training loss or the model accuracy.

Similar to any NN, PINNs have a number of hyperparameters such as, the learning rate, the activation function, the optimiser, the weights in the loss function, the width and depth of the NN. The high-dimensional hyperparameter search space makes it difficult to obtain optimised hyperparameters that are suitable for a broad range of problems. Therefore, it is necessary to select fewer hyperparameters and focus on a specific set of problems.

In this work, we investigated the neural network architecture, the number of hidden layers, the learning rate and the activation function over a wide range of settings, as detailed in Table 4.1.

Typically, hyperparameter optimisation is implemented using automated optimisation techniques to obtain the optimal settings. However, in this work, we employed a manual tuning approach [159] to obtain multiple settings that results in a relative L2 error of 10% or less across all five

Table 4.1: Range of hyperparameters explored.

Hyperparameters	Range
Architecture	FCNN, MFNN, DGM
No. of hidden layers	1, 5, 10, 15, 20
Activations	ReLU, SELU, SiLU, GELU, Tanh, Sin
Learning rates	1e-2, 7.5e-3, 5e-3, 2.5e-3, 1e-3, 7.5e-4, 5e-4, 2.5e-4, 1e-4, 7.5e-5, 5e-5, 2.5e-5, 1e-5

test cases. This approach was chosen to study the behaviour of the model across a wide range of parameters and to obtain a list of optimal settings that can be broadly applied, rather than identifying a single optimal configuration tailored to the specific problem at hand.

The weights were initialised with the Xavier initialisation [85] and the losses were calculated using the integral formulation (Equation 2.13). Initially, we experimented with activation functions such as rectified linear unit (ReLU), scaled exponential linear unit (SELU), Sigmoid linear unit (SiLU), Sine function (Sin), Gaussian error linear units (GELU) and hyperbolic tangent (Tanh) over a range of learning rates starting from 1e-5 to 1e-2. The mathematical form of the activation functions is listed below [160].

$$\begin{aligned}
\text{ReLU}(x) &= \max(0, x) \\
\text{SELU}(x) &= 1.0507 \begin{cases} x & \text{if } x > 0 \\ 1.6733(e^x - 1) & \text{if } x \leq 0 \end{cases} \\
\text{SiLU}(x) &= x \cdot \frac{1}{1 + e^{-x}} \\
\text{GELU}(x) &= 0.5x \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} (x + 0.044715x^3) \right) \right) \\
\text{Tanh}(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}}
\end{aligned} \tag{4.1}$$

where, x is the input value. It is important to note that the ReLU and SELU are non-differentiable. Most machine learning libraries handle the non-differentiable activation functions using subgradient optimisation. Bertoin et al. [161] emphasised that the derivative of non-differentiable activation functions can impact the backpropagation and the Adam optimiser can help minimise this impact. Therefore, we used the Adam optimiser in all test cases. The batch

training was performed on a high-performance computing (HPC) cluster, Sunbird at the Swansea University. The test cases were trained on 2 × AMD EPYC Rome 7452 processor with 32-cores and a NVIDIA A100 tensor core with 40GB of available GPU memory.

4.3 Test cases

The first four test cases were taken from Chapter 3, while the final test case was a steady-state heat conduction problem with discontinuous conductivity.

4.3.1 Problem 1 : 2D steady-state heat conduction

In Problem 1, we chose a 2D steady-state heat conduction problem with discontinuous BCs at the corners. The problem can be stated as follows:

$$\begin{aligned} u_{xx} + u_{yy} &= 0, & x \in [-0.5, 0.5], & y \in [0, 1] \\ u(x, 0) &= u(x, 1) = 0 & u(-0.5, y) &= u(0.5, y) = 1 \end{aligned} \quad (4.2)$$

where u is the temperature, x and y are the independent variables of the PDE.

4.3.2 Problem 2 : 3D steady-state heat conduction

Our next focus is on extending Problem 1 into 3D space. Problem 2 is constrained by discontinuous BCs at the edges and the corners. The problem is outlined as follows:

$$\begin{aligned} u_{xx} + u_{yy} + u_{zz} &= 0, & x \in [-0.5, 0.5], & y \in [-0.5, 0.5], & z \in [0, 1] \\ u(x, y, 0) &= u(x, y, 1) = u(0.5, y, z) = 0 \\ u(-0.5, y, z) &= u(x, -0.5, z) = u(x, 0.5, z) = 1 \end{aligned} \quad (4.3)$$

The reference solution for Problem 1 and 2 was generated through FEM implemented in MATLAB Partial Differential Equation Toolbox, is shown in Figure 4.1. Details of the mesh statistics of the FEM solutions is shown in Table 4.2.

The baseline PINN comprises a FCNN architecture with Tanh activation, trained using a learning rate of 1e-3. It was known to have convergence issues with discontinuous BCs, as

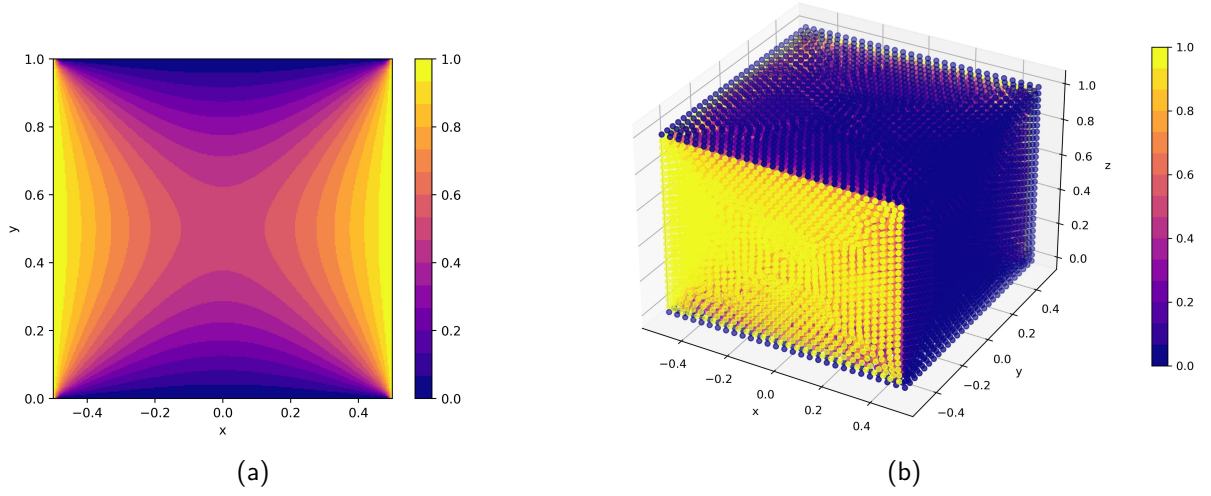


Figure 4.1: FEM Solution of Problem 1 (left) and Problem 2 (right).

Table 4.2: Mesh statistics of the FEM solution.

	Problem 1	Problem 2
Element type	triangle	tetrahedral
Element order	quadratic	quadratic
No. of elements	2841	23424
Max. element size	0.0566	0.0693
Min. element size	0.0283	0.0346
Mesh growth rate	1.5	1.5

demonstrated in the loss curve (Figure 4.2) of Problem 1, trained with a FCNN of 8 hidden layers and 20 units in each layer. Since, the baseline PINNs are continuous functions they cannot represent discontinuities. This is evident in Problem 1, where the pointwise BC loss (\mathcal{L}_{BC}) around the corners is relatively high after training for 14k iterations (Figure 4.3).

4.3.3 Problem 3 : Problem 1 with parametric geometry

In Problem 3, the upper boundary of Problem 1 is varying from 1 to 1.05 (Equation 4.4).

$$\begin{aligned}
 u_{xx} + u_{yy} &= 0, & x \in [-0.5, 0.5], & y \in [0, L], & L \in [1, 1.05] \\
 u(x, 0) &= u(x, L) = 0 & u(-0.5, y) &= u(0.5, y) = 1
 \end{aligned} \tag{4.4}$$

Parametric PDEs can be solved with a simple extension of the PINN by adding the parametric

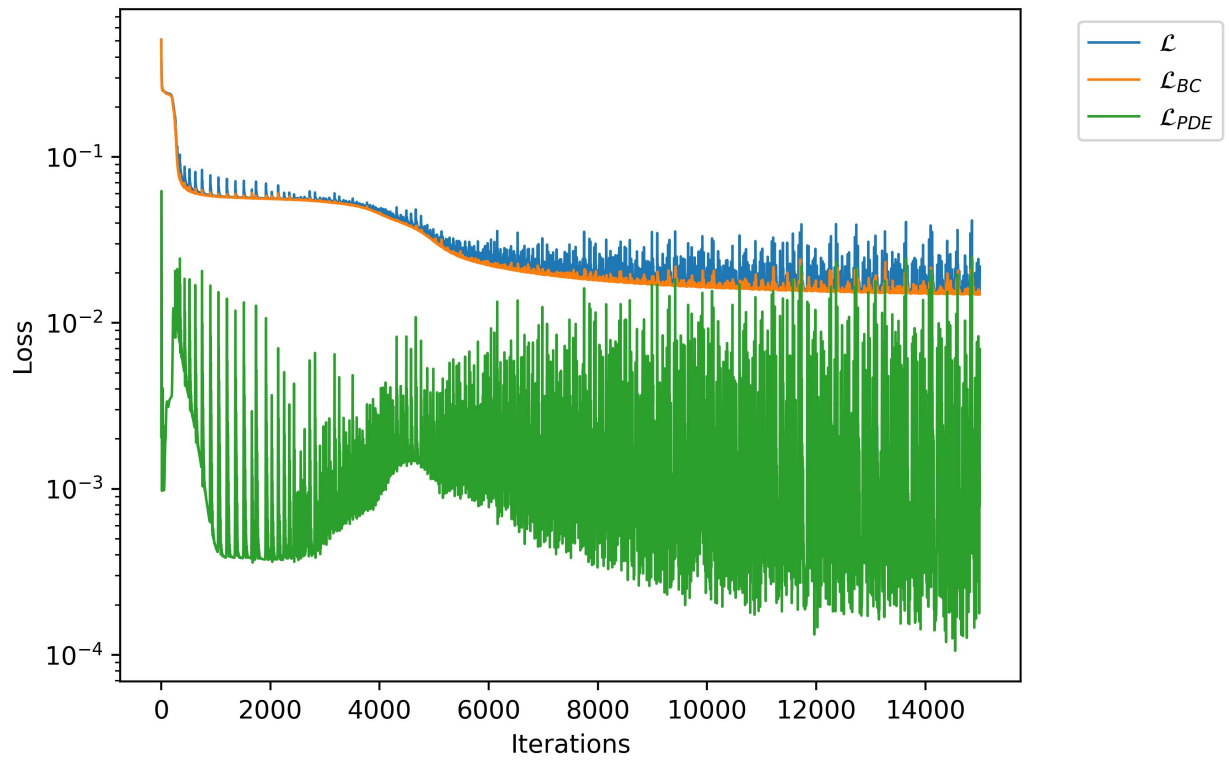


Figure 4.2: The loss curve (training loss) of Problem 1 trained with Tanh activation and baseline PINN for 14k iterations.

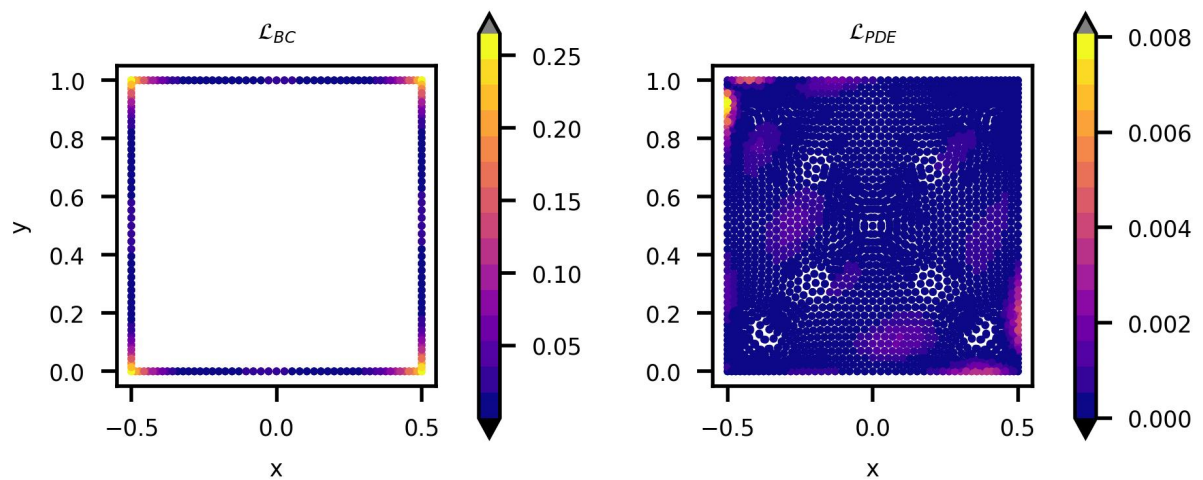


Figure 4.3: The pointwise BC loss (left) and pointwise PDE loss (right) of Problem 1 trained with Tanh activation and baseline PINN for 14k iterations.

terms as additional features in the training dataset. Since the upper boundary is not fixed, care should be taken to ensure the y-coordinate for each sample in the training dataset resides inside

the domain.

4.3.4 Problem 4 : Problem 1 with parametric conductivity

Similar to Problem 3 we can parameterise the conductivity. We formulate Problem 1 with parametric conductivity κ as follows.

$$\begin{aligned} u_{xx} + \kappa u_{yy} &= 0, \quad x \in [-0.5, 0.5], \quad y \in [0, 1], \quad \kappa \in [0, 1] \\ u(x, 0) &= u(x, 1) = 0 \quad u(-0.5, y) = u(0.5, y) = 1 \end{aligned} \quad (4.5)$$

4.3.5 Problem 5 : 2D steady-state heat conduction with discontinuous conductivity

In Problem 5, we have a 2D steady-state heat conduction problem with discontinuous conductivity κ and Robin BCs. The problem can be stated as follows:

$$\begin{aligned} \frac{\partial}{\partial x}(\kappa(x) \cdot u_x) + \frac{\partial}{\partial y}(\kappa(x) \cdot u_y) &= 0, \quad x \in [0, 1], \quad y \in [0, 1] \\ u_x &= \frac{u}{2\kappa(x)} \quad \text{when } x = 0, \quad u_x = \frac{1-u}{2\kappa(x)} \quad \text{when } x = 1 \\ \kappa(x) &= \begin{cases} \kappa_1 & \text{if } x \leq 0.5 \\ \kappa_2 & \text{if } x > 0.5 \end{cases} \end{aligned} \quad (4.6)$$

where $\kappa_1 = 1/30$ and $\kappa_2 = 1/15$. Here, the pointwise conductivity varies as a piecewise function of the x -coordinates. The problem has an analytical solution given as a piecewise function of κ_1 and κ_2 (Equation 4.7) [162].

$$u = \begin{cases} \frac{\kappa_2 x + 2\kappa_1 \kappa_2}{0.5(\kappa_1 + \kappa_2) + 4\kappa_1 \kappa_2} & \text{if } x \leq 0.5 \\ \frac{\kappa_1 x + 2\kappa_1 \kappa_2 + 0.5(\kappa_2 - \kappa_1)}{0.5(\kappa_1 + \kappa_2) + 4\kappa_1 \kappa_2} & \text{if } x > 0.5 \end{cases} \quad (4.7)$$

4.4 Results and discussion

In this section, we present the results of hyperparameter optimisation across all the test cases. We leverage bar charts to discuss the relationship between the hyperparameters and the mean

relative L2 error for all possible settings. These bar charts illustrate the mean relative L2 error for each hyperparameter to identify the problematic hyperparameter values.

SELU is known for fast and stable training, however, based on the initial investigation, we observed that the SELU activation function is causing convergence issues in all the test cases. This is due to the fact that SELU is prone to exploding gradients in deep networks [163].

Unlike SELU, RELU suffers from vanishing gradients. So even though the training loss is converged, it exhibits a relative L2 error that is several magnitudes higher compared to other activation functions across most test cases. This is due to the fact that, for negative inputs, the gradient of RELU is zero. During backpropagation, a large negative gradient multiplied with zero effectively vanishes the gradient. So, the corresponding neurons will not be updated properly, resulting in a high relative L2 error. Thus, we decided to exclude SELU and RELU activation from further studies and continued further investigation with GELU, SiLU, Sin and Tanh.

Figure 4.4, shows the mean relative L2 error for each activation across all the test cases. Notable anomalies were observed in Problem 3, where the mean relative L2 error for Tanh activation reached the order of 10^5 , and for Sin activation exceeded 10^0 (100%). These anomalies are attributed to configurations using a large learning rate 10^{-2} .

Figure 4.5, illustrates the relationship between the mean relative L2 error with number of hidden layers. As anticipated, the mean relative L2 error decreases with the increase in the number of hidden layers. With the addition of layers the network gains the ability to learn more complex patterns. Similar to activations, in Problem 3, with 5 hidden layers, the mean relative L2 error reaches to the order of 10^5 . This anomaly is attributed to all the settings with large learning rate 10^{-2} .

In Figure 4.6, we present the variation of mean relative L2 error with neural network architectures. Similar to the activations and the number of hidden layers, in Problem 3, the mean relative L2 error has been skewed by an anomaly attributed to the DGM architecture and the use of a large learning rate. It becomes evident that large learning rates may lead to anomalies or unexpected behaviour and must be avoided while solving stiff-PDEs.

Although, we used SDF in all the settings, it is worth noting that FCNN has a lower mean relative L2 error than MFNN in all the test cases. The SDF excludes the regions with high-frequency function in the domain, thus FCNN is less susceptible to spectral bias. On the other hand, MFNN projects the low-dimensional input dataset to higher dimensions thus increasing the complexity of the model. Thus, MFNN may need more iterations to converge, potentially affecting the performance compared to FCNN.

Figure 4.7 shows the variation of mean relative L2 error with learning rate. In Problem 3,

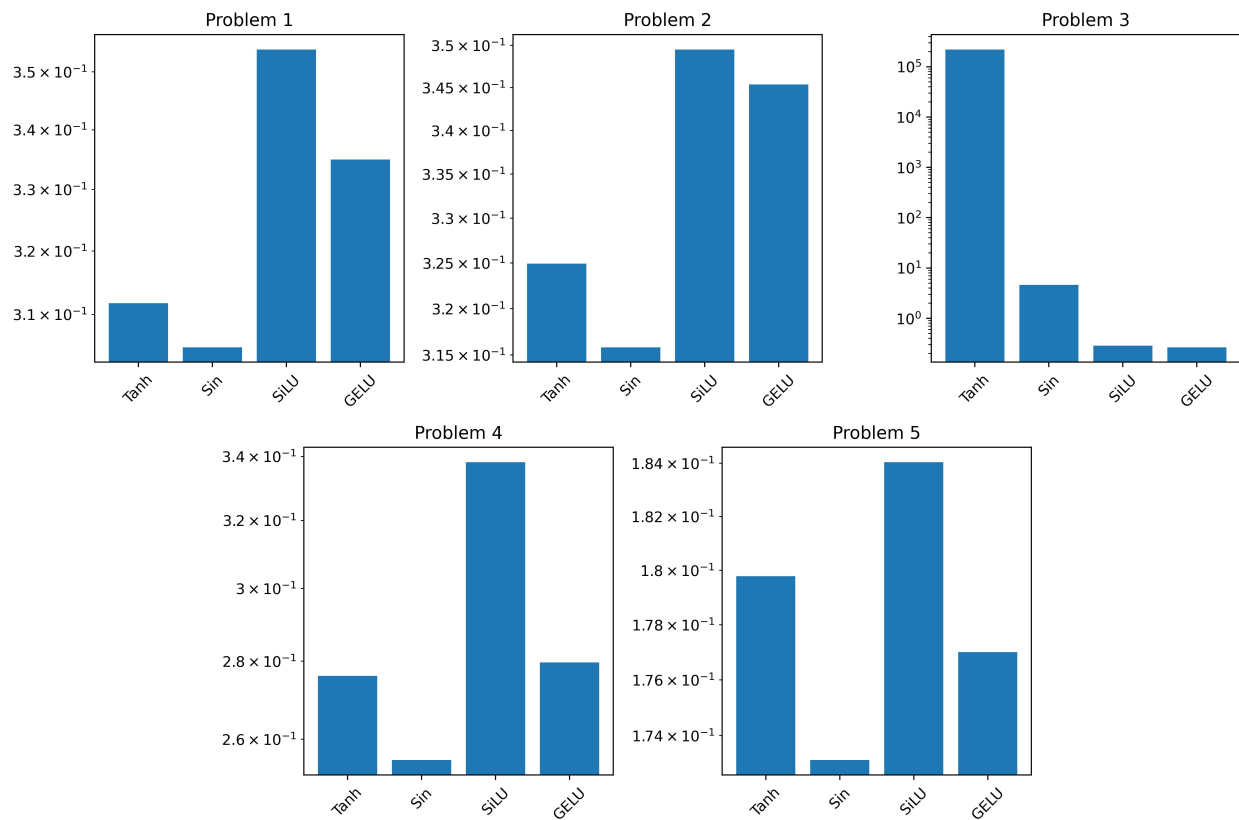


Figure 4.4: Comparison of activation function for five different test cases. The horizontal axis shows the activation function while the vertical axis shows the mean relative L2 error.

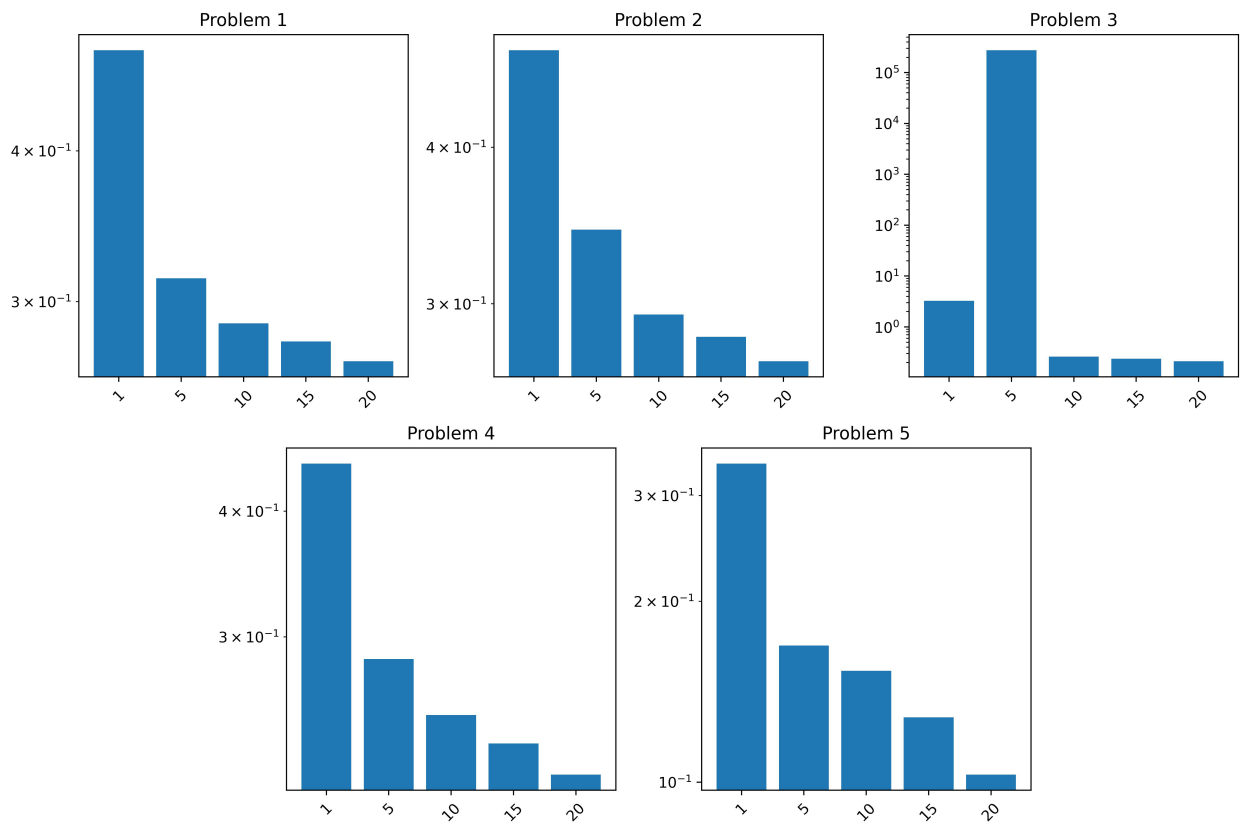


Figure 4.5: Comparison of number of hidden layers for five different test cases. The horizontal axis shows the number of hidden layers while the vertical axis shows the mean relative L2 error. For each specific number of hidden layers, the mean relative L2 error includes data from all architectures with that specific number of hidden layers.

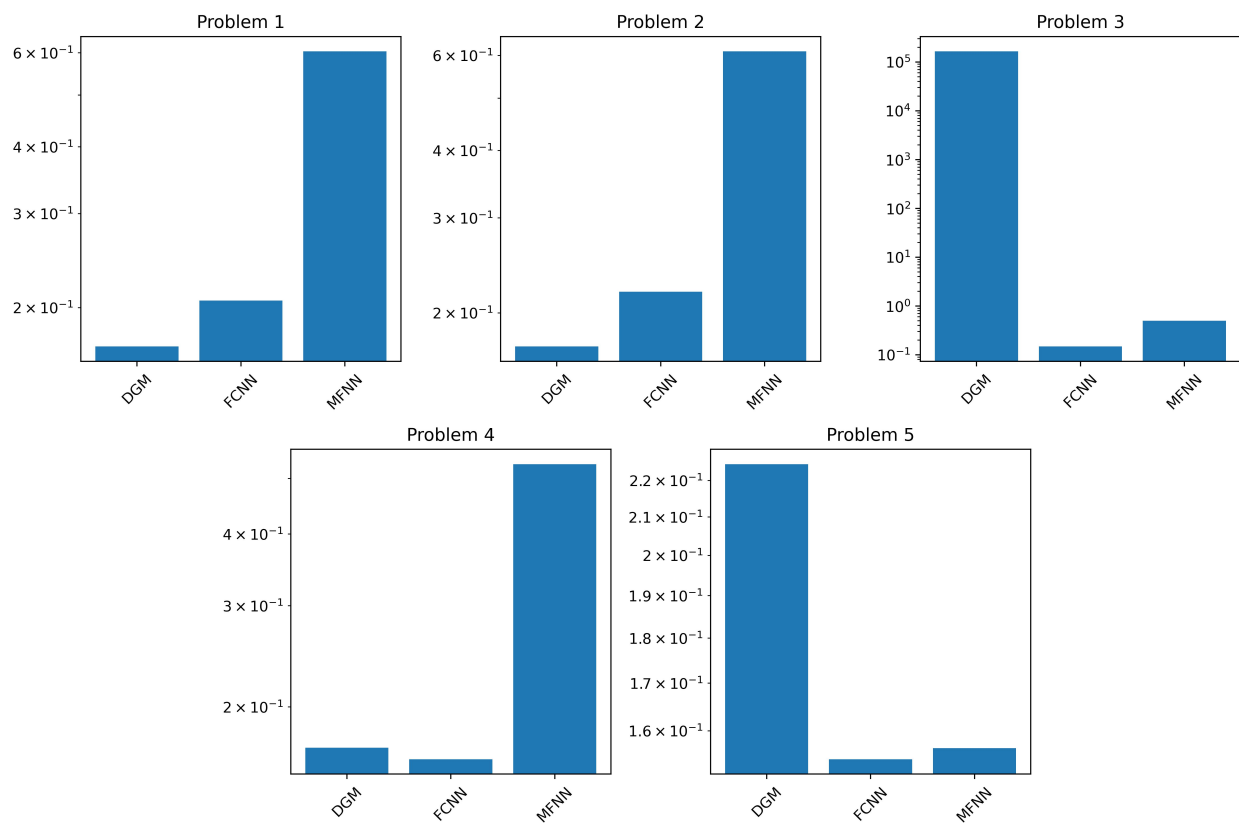


Figure 4.6: Comparison of neural network architectures for five different test cases. The horizontal axis shows the individual neural network architectures while the vertical axis shows the mean relative L2 error.

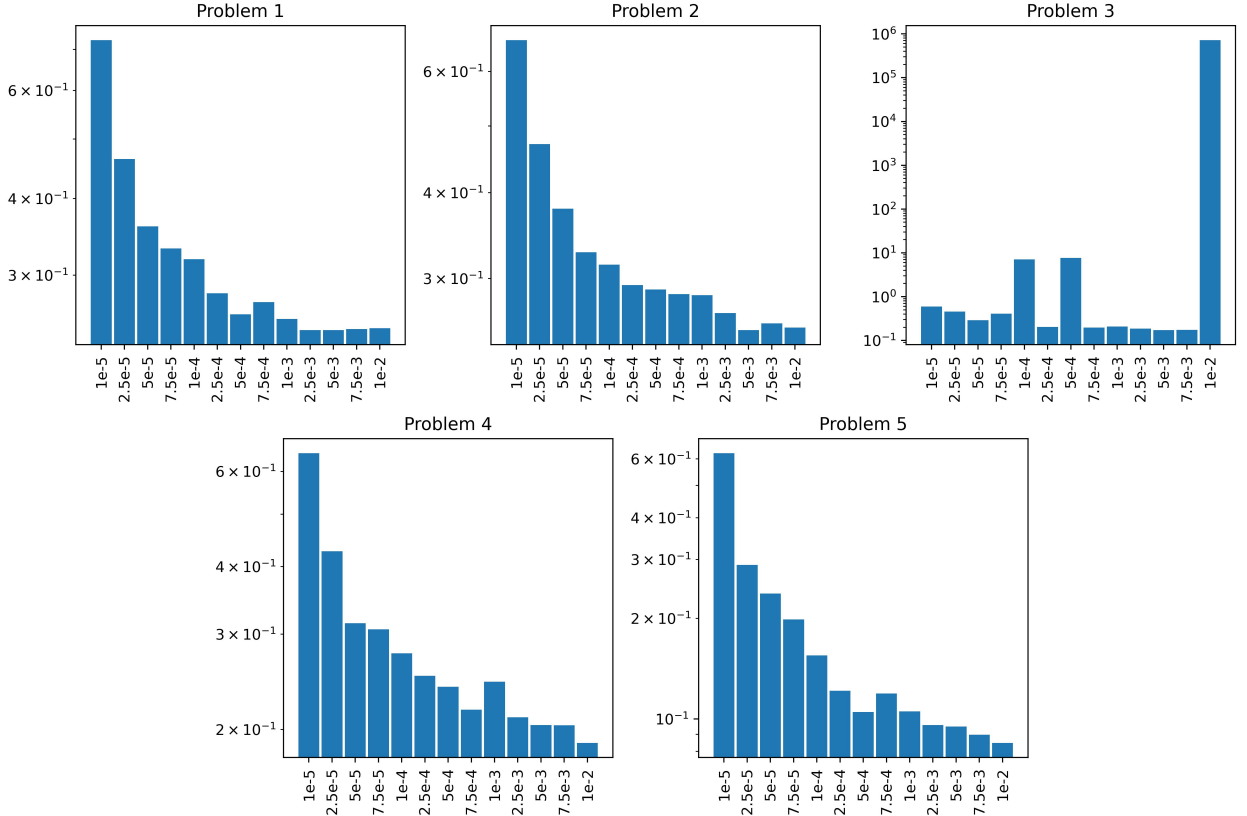


Figure 4.7: Comparison of learning rates for five different test cases. The horizontal axis shows the learning rates while the vertical axis shows the mean relative L2 error.

all the anomalies discussed earlier were caused by the learning rate of 10^{-2} . This is due to the parametric nature of Problem 3, where the moving discontinuity around the upper boundary results in a highly complex loss landscape. Consequently, the optimiser fails to find the global minima, resulting in a mean relative L2 error of the order of 10^6 .

In contrast, for all other test cases, we observed an improvement in the mean relative L2 error with an increase in learning rate. This is a well-known trend with the Adam optimiser where the learning rate determines the step size taken in each iteration. However, due to inherent highly non-convex nature of PINNs, the optimiser may get trapped in a local minima when using small learning rates.

After analysing the data, we have identified optimal hyperparameter settings, these are shown in Table 4.3. With these settings, we obtained a mean relative L2 error of 5.60%, a maximum relative L2 error of 13.9% with a standard deviation of 1.08% across all the test cases. While our optimal number of hidden layers was 20, it should be noted that more hidden layers can be used. However, in such cases, the number of iterations should be enough to allow for convergence. Furthermore, Figure 4.8 shows the distribution of relative L2 error across these test cases. It

Table 4.3: Optimal hyperparameter settings.

Hyperparameters	Range
Architecture	FCNN, DGM
No. of hidden layers	20
Activations	SiLU, GELU, Tanh, Sin
Learning rates	7.5e-3, 5e-3, 2.5e-3, 1e-3, 7.5e-4, 5e-4

demonstrates that in majority of the test cases the relative L2 error remains below 10%.

Alternatively, if we restrict the list of optimal activations to SiLU, we achieved a mean relative L2 error of 5.32%, a maximum relative L2 error of 9.55% with a standard deviation of 0.7048% across all the test cases. This is in agreement with the objective of the work i.e., the optimal settings should achieve a relative L2 error of 10% or less across all the test cases. However, we observed that, on case-by-case basis, other activations performed better than SiLU. Therefore, we include other activations in the list of optimal activations, allowing the reader to choose the most suitable activation function for specific scenarios.

4.5 Conclusions

In this study, we conducted a comprehensive hyperparameter optimisation of PINNs to determine the optimal neural network architectures, number of hidden layers, activation functions and learning rates for heat conduction problems with discontinuous solutions. We aimed to identify all possible settings of these hyperparameters that could achieve a relative L2 error of 10% or less across all test cases.

These test cases include a 2D and a 3D steady-state heat conduction problem with discontinuous BCs. Additionally, we extended the 2D steady-state heat conduction problem with parametric conductivity and parametric geometry. Finally, the last test case involved a 2D steady-state heat conduction problem with discontinuous conductivity and Robin BCs.

Throughout the investigation, we employed a range of tools, such as, adaptive activation, SDF, importance sampling and quasi-random sampling. These methods accelerate the convergence of the training loss, resulting in much lower relative L2 error compared to not using them. Specifically, the SDFs have limited ability to accurately capture discontinuities in the solution. As a result, models with a relative L2 error within 10% are generally considered to be good models.

The range of hyperparameters investigated has been summarised in Table 4.1. In Section 4.4, we discussed the relationship between each hyperparameter and the mean relative L2 error. Finally, we presented a list of optimal hyperparameter settings in Table 4.3.

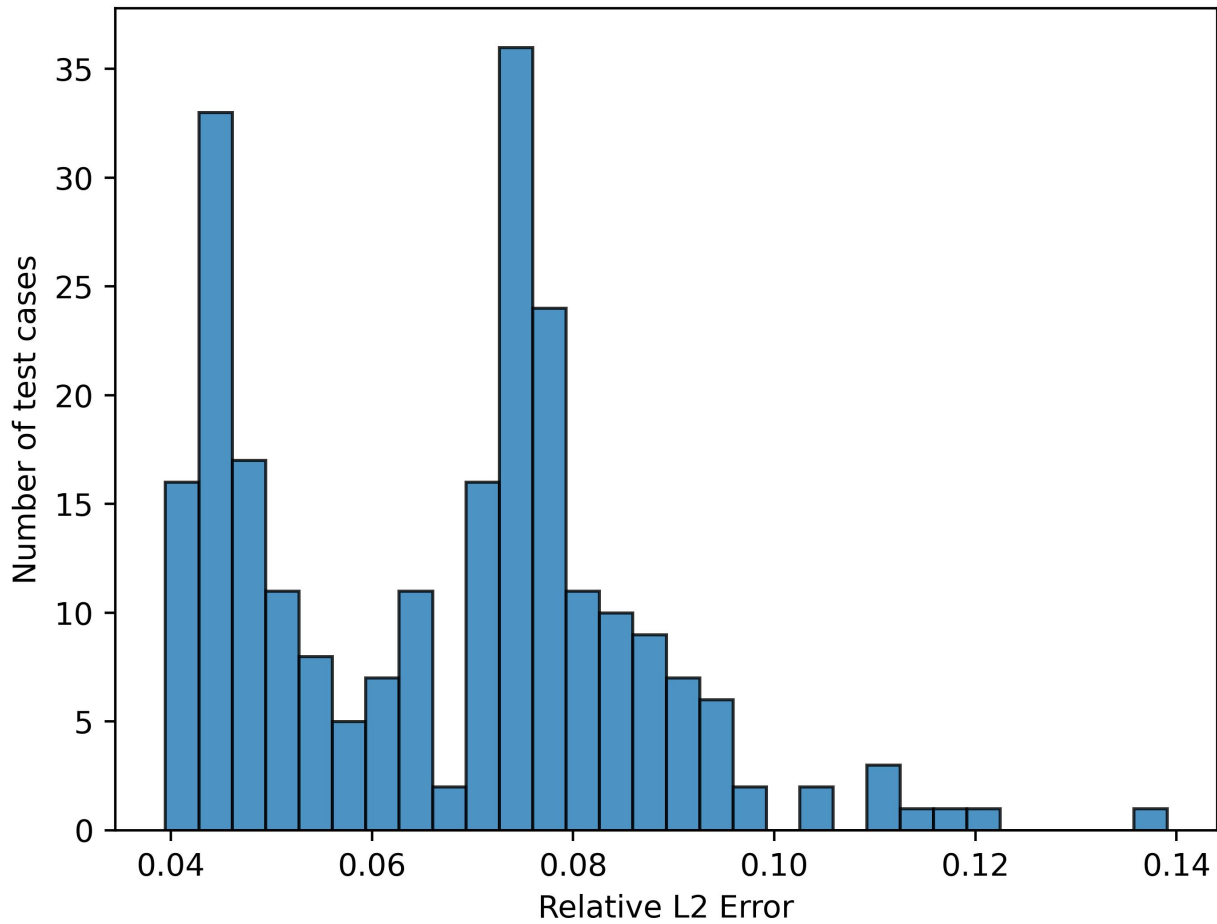


Figure 4.8: The distribution of relative L2 error with optimised settings across all the test cases.

In general, hyperparameters that are found to be effective for a specific class of problems can often be used for similar problems. Therefore, these optimal hyperparameters can be used directly to obtain a satisfactory solution i.e. a relative L2 error less than 10% and as a starting point for obtaining optimal hyperparameters for other heat conduction problems that exhibit similar discontinuities. Examples of variations may include changes in the geometry, different boundary temperatures, and the introduction of source terms in the PDE. However, the effectiveness of these optimal hyperparameters must be validated on a case-by-case basis.

The hyperparameter optimisation of PINNs is an effective approach to obtain accurate solutions for heat transfer problems with discontinuous solution. This work also demonstrates the potential of hyperparameter optimisation to enhance the predictive capability of PINNs for solving stiff-PDEs.

Chapter 5

Inverse solution reconstruction

5.1 Introduction

Inverse modelling is an important area in science and engineering. In contrast to forward problems, inverse problems involve the estimation of the input parameters of a forward problem. These problems are ill-posed, potentially leading to multiple solutions [164]. One of the earliest approaches to solve inverse problems involved a brute-force parameter space search [165, 166]. However, for complex problems that involve a large number of parameters, the search is combinatorially intractable or requires an excessively long time. Alternatively, an inverse problem can be reformulated as an approximate well-posed problem by incorporating prior knowledge of the specific problem [167]. A standard approach is to redefine the ill-posed inverse problem as an optimisation problem, where regularisation techniques are applied to integrate the prior knowledge. This approach enables the solution of complex problems within a reasonable time frame [168].

Inverse problems play a crucial role in the field of heat transfer. A number of inverse problems have been discussed in [169]. These problems include estimation of the strength and location of the heat source in heat treatment processes, the wall temperature distribution in a furnace and the transport in a plume or jet using Lagrange multipliers [170]. Instances of employing particle swarm optimisation (PSO) to estimate the heat source on a vertical flat plate with natural convection can be found in [171]. In addition, Yaman *et al.* [172] reviewed existing algorithms for parameter estimation. They summarised algorithms aiming to find locations, shapes, and boundary parameters of obstacles.

In recent times, with the advent of deep learning, it is possible to fit the sparse measurements on a highly nonlinear trial function called a deep neural network (DNN). A DNN with as few as one hidden layer is a universal approximator [173] regardless of the activation function [174].

DNNs do not require explicit programming and they can provide an accurate inverse estimate of parameters within a reasonable time frame. For example, the authors in References [175, 176] used convolutional neural networks (CNN) to inversely estimate several parameters in thermal and fluid flow problems. Tamaddon-Jahromi *et al.* [177] developed a data-driven feed-forward neural network model to inversely predict the BCs for linear and non-linear heat conduction, convection-conduction, and natural convection problems.

As mentioned in an earlier chapter, PINNs [20] have gained popularity due to their novelty in solving forward [21, 23, 24, 96, 99, 147] and inverse problems [18, 26, 27, 178, 179] involving PDEs using neural networks (NN). Unlike conventional numerical techniques for solving PDEs, PINNs represent a hybrid approach, leveraging both data-driven and physics-based methods. PINNs inherently satisfy the initial condition (IC), BCs, and the governing PDE, enabling them to solve forward problems. For inverse problems, where one or more of these elements may be unknown or unclear, PINNs can learn the missing information from sparse measurements. The reader is referred to reviews on PINNs including a survey on applications to problems with discontinuous BCs [150, 180].

Applications of PINNs have been effectively extended to inverse problems, particularly in the estimation of PDE coefficients such as mass, conductivity, and viscosity, leading to a series of works in many fields, including, fluids, solids, biomechanics, chemical reactions, etc. [91, 102, 181–183]. There have also been instances of applying PINNs to solution reconstruction, another category of inverse problems. Raissi *et al.* [184] employed measurement data, along with known PDEs and BCs, to reconstruct the solution for the flow past a cylinder problem. They utilised between 1,500 to 15,000 spatial measurements per time slice to achieve reasonable accuracy. This poses a significant challenge for real-world applications due to the extensive number of sensors deployed.

In a significant contribution, Bard *et al.* [88] employed a transformer PINN [73] to reconstruct the full magnetohydrodynamic solutions from sparse measurements, with known PDEs and BCs. The transformer PINN required 8-9 hours of training with 300 sparse measurements positioned around the boundaries, but it does not seem to scale well to higher dimensions. In a similar work, [185] used a large number of numerical solution as inputs to the PINN alongside the regular inputs to predict the BCs in real-time. A variety of works have focused on solution reconstruction in solids, fluids, and biomechanics [186–188]. These approaches may not be suitable in scenarios that demand the reconstruction of field variables without explicit knowledge of the BCs, while still utilising a reasonable number of sparse measurements.

In this chapter, we introduce a generalised workflow for reconstructing field variables that

addresses critical challenges faced in the application of PINNs to inverse problems. Unlike purely data-driven models, which necessitate extensive pre-training on large datasets, PINNs primarily face the challenge of requiring the deployment of an impractical number of sensors in experimental setups. This issue, along with the reliance on explicit knowledge of BCs, significantly limits their applicability to inverse problems such as solution reconstruction.

The proposed method offers a solution by leveraging sparse measurements to predict the BCs through an inverse workflow. These predicted BCs and the known PDE are used as input in a forward workflow to reconstruct the field variables. A new loss term ‘BC difference’ is specifically designed to fit smooth functions on the predicted BCs, thereby enhancing control over these BCs. Additionally, a ‘bound’ loss term is also incorporated to ensure that these predicted BCs remain within a specified bound. Consequently, the workflow requires the knowledge of sparse measurements, the governing PDE and expected ranges for each BC. By incorporating these elements, the workflow transforms an ill-posed inverse problem to an approximate well-posed problem.

The remainder of this chapter is organised as follows. Section 5.2 discusses the proposed workflow. In Section 5.3, we discuss network architecture and key parameters such as the activation function, learning rate, optimiser, and the computational setup employed in this study. It also presents the reconstructed temperature field in two distinct 2D steady-state heat conduction problems, each characterised by unique boundary conditions, and a 2D problem governed by the Helmholtz equation. Finally, we discuss the limitations and potential improvements for the proposed workflow in Section 2.10.

5.2 The proposed workflow

In this section, we discuss the overall structure and the functioning of the proposed workflow, which comprises both inverse and forward sections. Figure 5.1 shows a schematic diagram of the proposed workflow. The sparse measurements are fed into the inverse workflow, which predicts the BC on all boundaries. Then, the predicted BCs and the known PDE are used to reconstruct the solution using a forward workflow. In this chapter, the baseline PINN was used as a forward workflow. In Sections 5.2.1 and 5.2.2, detailed discussion of the inverse and forward workflows are provided, respectively.

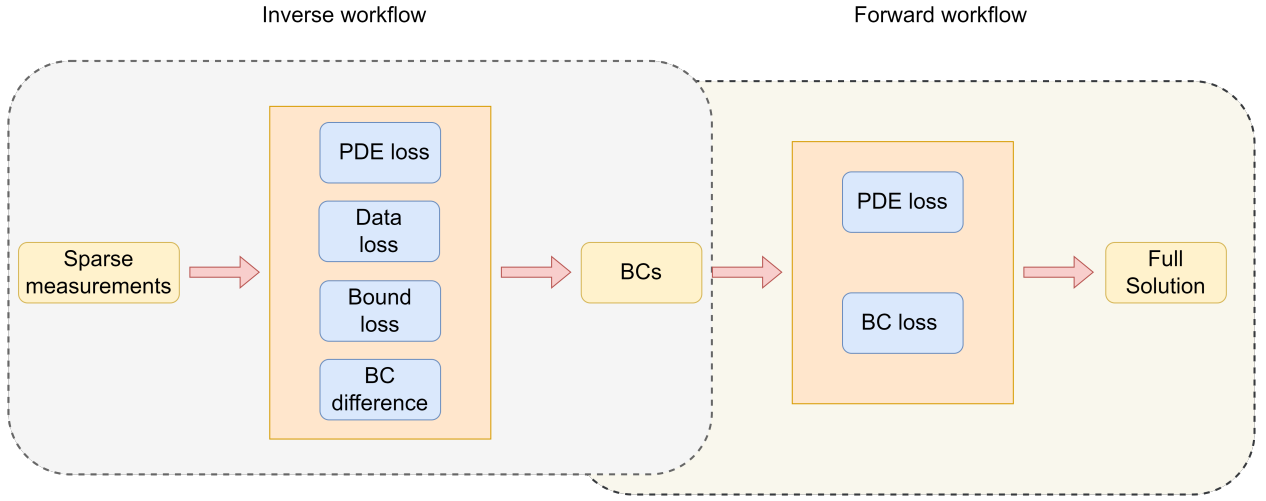


Figure 5.1: Schematic diagram of the proposed workflow. The inverse workflow is on the left and the forward workflow is on the right.

5.2.1 Inverse workflow

This section discusses the details of the inverse workflow. Consider an ill-posed inverse problem as follows:

$$\begin{aligned} \mathbf{u}_t + \mathcal{N}_x[\mathbf{u}] &= 0, \quad \mathbf{x} \in \Omega, t \in [0, T] \\ \mathbf{u}_{true}(\mathbf{x}, t) &= \mathbf{u}(\mathbf{x}, t)[N_d], \quad \mathbf{x} \in \Omega, t \in [0, T] \end{aligned} \quad (5.1)$$

where $\mathcal{N}_x[\mathbf{u}]$ is a differential operator, \mathbf{x} and t are the independent variables of the PDE, Ω denotes the spatial domain, T is the final time and \mathbf{u}_{true} are the sparse measurements.

In this chapter, we assume that N_d sparse measurements are sampled, either randomly or through other methods, from the known FEM solution $\mathbf{u}(\mathbf{x}, t)$. Sparse measurements \mathbf{u}_{true} obtained from the experiments may contain noise. We assume that the noise in our sparse measurements \mathbf{u}_{true} is white noise.

In the literature surrounding PINNs, the IC is treated as a subset of the BCs. This is because the IC is prescribed at the initial time, similar to how BCs are imposed on the boundary of the spatial domain.

In the context of PINNs for inverse modelling, the traditional loss function consists of three main components: \mathcal{L}_{PDE} , \mathcal{L}_{Data} , and \mathcal{L}_{BC} . Here, \mathcal{L}_{PDE} ensures the solution satisfies the governing PDEs, \mathcal{L}_{Data} quantifies the discrepancy between the NN predictions and the observed sparse measurements, and \mathcal{L}_{BC} enforces the solution's adherence to the prescribed BCs. Typically, the

total loss function is expressed as follows:

$$\mathcal{L}_{inverse} = \lambda_{PDE} \mathcal{L}_{PDE} + \lambda_{Data} \mathcal{L}_{Data} + \lambda_{BC} \mathcal{L}_{BC} \quad (5.2)$$

where λ_{PDE} , λ_{Data} , and λ_{BC} are the weights for the respective loss components. This framework is further refined to enhance its applicability to specific challenges in inverse modelling such as parameter estimation and solution reconstruction, as discussed in Section 5.1.

However, the solution reconstruction becomes extremely challenging when the BCs are partially known or poorly defined. To address these challenges, this work incorporates extra information through prior knowledge, such as the approximate range of the solution, and applies this knowledge as L2 regularisation within the loss function. This approach enables a more robust solution to inverse problems by leveraging available information to guide the solution process, even in the absence of well-defined BCs.

Two new loss terms have been introduced: bound loss (\mathcal{L}_{Bound}) and BC difference loss ($\mathcal{L}_{BC_{Diff}}$). The \mathcal{L}_{Bound} ensures that the solution remains within a specified range, while $\mathcal{L}_{BC_{Diff}}$ aligns smooth functions with the predicted BCs. Detailed discussions of these terms are provided in sections 5.2.1.1 and 5.2.1.2. The total loss, incorporating the PDE loss and the data loss, are defined as follows:

$$\mathcal{L}_{inverse} = \lambda_{PDE} \mathcal{L}_{PDE} + \lambda_{Data} \mathcal{L}_{Data} + \lambda_{Bound} \mathcal{L}_{Bound} + \lambda_{BC_{Diff}} \mathcal{L}_{BC_{Diff}} \quad (5.3)$$

$$\mathcal{L}_{PDE} = \frac{1}{N_r} \sum_{i=1}^{N_r} |\hat{u}_t(\mathbf{x}_i, t_i) + \mathcal{N}_x[\hat{u}(\mathbf{x}_i, t_i)]|^2 \quad (5.4)$$

$$\mathcal{L}_{Data} = \frac{1}{N_d} \sum_{i=1}^{N_d} |\hat{u}(\mathbf{x}_i, t_i) - u_{true}(\mathbf{x}_i, t_i)|^2 \quad (5.5)$$

where \hat{u} represents the predicted solution from the inverse workflow, N_r is the number of collocation points that satisfies the residual of the PDE. The data loss (\mathcal{L}_{Data}) accounts for the N_d sparse measurements u_{true} . The coefficients λ_{PDE} , λ_{Data} , λ_{Bound} and $\lambda_{BC_{Diff}}$ in Equation 5.3, help to balance the contribution of \mathcal{L}_{PDE} , \mathcal{L}_{Data} , \mathcal{L}_{Bound} and $\mathcal{L}_{BC_{Diff}}$ in the inverse workflow.

5.2.1.1 Bound loss (\mathcal{L}_{Bound})

The Bound loss (\mathcal{L}_{Bound}) not only ensures that the predicted BCs remain within a specified bound defined by its lower (\mathcal{A}_{lb}) and upper (\mathcal{A}_{ub}) bounds, but it can also be used to constrain the

reconstructed solution itself, depending on the amount of knowledge we have about the specific experiment or problem. For instance, the bound loss (\mathcal{L}_{Bound}) to constrain the reconstructed solution is defined as follows:

$$\mathcal{L}_{Bound} = \text{ReLU}(\hat{u} - \mathcal{A}_{ub}) + \text{ReLU}(\mathcal{A}_{lb} - \hat{u}) \quad (5.6)$$

where ReLU is rectified linear unit, as detailed in Equation 5.7, Figure 5.2 shows the $\text{ReLU}(z)$ given input z . For any input z less than zero the $\text{ReLU}(z)$ is zero. We leverage this property in our formulation of the bound loss (\mathcal{L}_{Bound}). The bound loss ensures that the loss is zero when the predicted solution \hat{u} is within the bounds \mathcal{A}_{lb} and \mathcal{A}_{ub} ; outside these bounds, the error increases linearly.

$$\text{ReLU}(z) = \max(0, z) \quad (5.7)$$

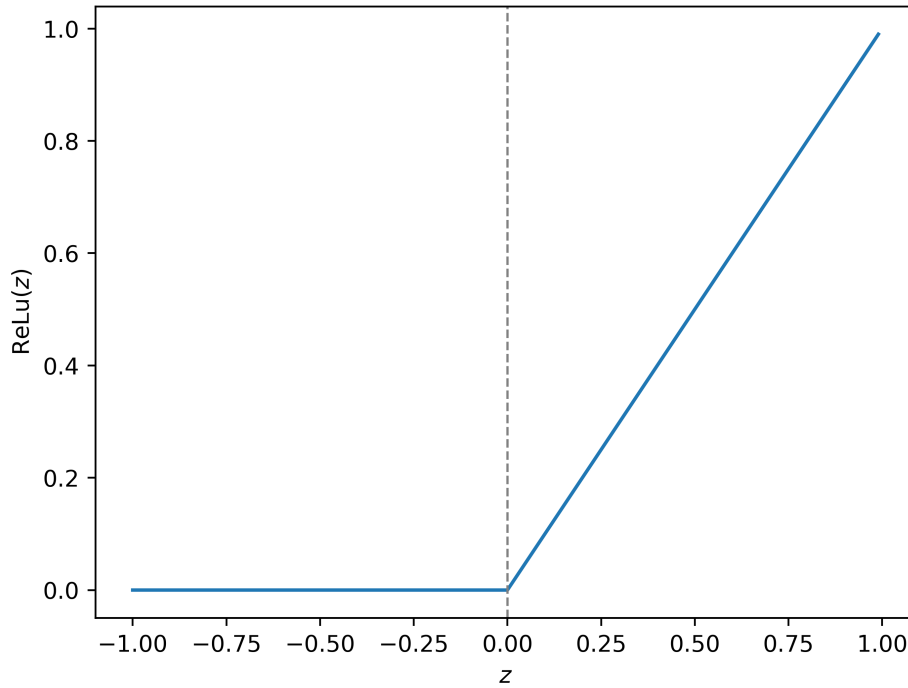


Figure 5.2: The ReLU function.

Equation 5.6 can also be used to constrain the BCs, whether Dirichlet or Neumann, by replacing \hat{u} with predicted BCs. Depending on the available experimental knowledge \mathcal{A}_{lb} and \mathcal{A}_{ub} can represent a common range of all Dirichlet and Neumann BCs, either independently or in mixed form.

5.2.1.2 BC difference loss ($\mathcal{L}_{BC_{Diff}}$)

The BC difference loss ($\mathcal{L}_{BC_{Diff}}$) is employed to align smooth functions, a function having continuous derivatives up to a certain order, with the predicted BCs. Initially, it is necessary to determine the type of BC (Dirichlet or Neumann) for all boundaries. Based on this classification, the loss function $\mathcal{L}_{BC_{Diff}}$ may be expressed as follows:

$$\mathcal{L}_{BC_{Diff-Dirichlet}} = \sum_{i=1}^D \sum_{j=1}^{N_{BC}^i} \frac{1}{N_{BC}^i} |\hat{u}_i(\mathbf{x}_j, t_j) - \mathcal{G}_i(\mathbf{x}_j, t_j)|^2 \quad (5.8)$$

$$\mathcal{L}_{BC_{Diff-Neumann}} = \sum_{i=1}^N \sum_{j=1}^{N_{BC}^i} \frac{1}{N_{BC}^i} \left| \frac{\partial \hat{u}_i}{\partial n} \Big|_{(\mathbf{x}_j, t_j)} - \mathcal{G}_i(\mathbf{x}_j, t_j) \right|^2 \quad (5.9)$$

$$\mathcal{L}_{BC_{Diff}} = \mathcal{L}_{BC_{Diff-Dirichlet}} + \mathcal{L}_{BC_{Diff-Neumann}} \quad (5.10)$$

where D is the number of Dirichlet boundaries, N is the number of Neumann boundaries, N_{BC}^i is the number of points sampled on the i th boundary, \mathcal{G} is a smooth function of the independent variables. The term $\frac{\partial \hat{u}_i}{\partial n}$ represents the derivative of the inverse workflow's output, \hat{u}_i , in the normal direction at the boundary. The derivative $\frac{\partial \hat{u}_i}{\partial n}$ can be computed using automatic differentiation, a tool that automates the differentiation of highly complex functions, such as neural networks [30].

The smooth function (\mathcal{G}) in Equation 5.8 and 5.9, plays a key role in regulating the behaviour of the predicted BCs. For example, in cases where a BC is precisely known, such as an insulated BC, it can be directly implemented. In other scenarios, smooth functions can be effectively used to approximate the BCs. In this work, we assume that the BCs are independent of time. Therefore, we employed a polynomial function of order n as a smooth function $\mathcal{G}(\mathbf{x})$ i.e.,

$$\mathcal{G}(\mathbf{x}) = \sum_{k=0}^n a_k \mathbf{x}^k \quad (5.11)$$

where a_k for $k = 0, 1, 2, \dots, n$ are the external trainable parameters of the inverse workflow. These parameters will change their value over the course of training to minimise the BC difference loss (Equation 5.10). In scenarios where the BCs are time-dependent, a more sophisticated smooth function $\mathcal{G}(\mathbf{x}, t)$ may be needed.

While implementing the polynomial function $\mathcal{G}(\mathbf{x})$, it was observed that selecting a higher-order polynomial function for $\mathcal{G}(\mathbf{x})$, when the BCs might only require a lower order for an accurate fit, could significantly prolong the training process or even results in overfitting. To address this issue, we explored sparsity promoting techniques, detailed in the next section.

5.2.1.3 Promoting sparsity in polynomial function $\mathcal{G}(x)$

By nature Equation 5.11 has the potential for overfitting and prolonging training times when higher-order polynomial functions $\mathcal{G}(x)$ are used for BCs that require simpler representation. To mitigate these challenges, an algorithm was developed which promotes sparsity in the polynomial function $\mathcal{G}(x)$. This algorithm reduces complexity by identifying and minimising the influence of non-essential terms in $\mathcal{G}(x)$.

A key inspiration for the sparsity-promoting algorithm comes from the sequential thresholded least-squares algorithm of the SINDy framework, originally developed by [189] for identifying sparse governing equations in dynamical systems.

In Algorithm 5.1, a method was implemented to selectively 'turn off' the training of certain coefficients a_k in $\mathcal{G}(x)$ for each boundary. First, a sparsity knob or a predefined threshold is assumed. During training, if the magnitude of a specific coefficient remains below the sparsity knob for certain number of training iterations, that coefficient is then excluded from further training. This technique prevents unnecessary model complexity and computational overhead.

5.2.1.4 Sparsity check algorithm

We developed Algorithm 5.1 to promote sparsity in the polynomial function $\mathcal{G}(x)$ (See Equation 5.11). We first initialise an array of coefficients *coeff*, representing the coefficients a_k for $k = 0, 1, 2, \dots, n$ for each boundary. Accompanying each coefficient, a corresponding *sparsity_list* is defined, which is initially set to *False* for all entries, indicating that no coefficients have yet met the criteria for sparsity.

During the training, the algorithm updates *sparsity_list* by setting individual entries to *True* once they fulfil the sparsity conditions. These conditions are met when a coefficient's magnitude consistently remains below a predefined tolerance *sparsity_knob*, for a specified number of training iterations defined by *Sparsity_Threshold_Duration*.

For each coefficient, the number of consecutive iterations (*Below_Threshold_Count*) during which its magnitude remains below the *sparsity_knob* is tracked. If *Below_Threshold_Count* exceeds the *Sparsity_Threshold_Duration* then we set the coefficient as sparse in the *sparsity_list* to *True* and disable the gradient computation. This effectively excludes that coefficient from further training and thus promotes sparsity.

Algorithm 5.1 : Promoting sparsity in polynomial function $\mathcal{G}(x)$ (Equation 5.11)

```

1: Input:
2:   coeff: Array of coefficients, representing  $a_k$  for  $k = 0, 1, 2, \dots, n$ .
3:   sparsity_list: Boolean array, each element denotes if the corresponding coefficient in coeff is considered sparse.
4:   sparsity_knob: Tolerance level for sparsity, a predefined threshold value.
5:   Below_Threshold_Count: Array of counters, each tracking the number of consecutive iterations a coefficient's magnitude remains below sparsity_knob.
6:   Sparsity_Threshold_Duration: The number of consecutive iterations below sparsity_knob required to consider a coefficient as sparse.
7: Output:
8:   Modified sparsity_list.
9: procedure SPARSITY_CHECK(coeff, sparsity_list, sparsity_knob, Below_Threshold_Count, Sparsity_Threshold_Duration)
10:  for  $i \leftarrow 0$  to  $\text{length}(\text{coeff}) - 1$  do
11:    if  $|\text{coeff}[i]| < \text{sparsity\_knob}$  and  $\text{sparsity\_list}[i] = \text{False}$  then
12:      Below_Threshold_Count[ $i$ ]  $\leftarrow$  Below_Threshold_Count[ $i$ ] + 1
13:      if Below_Threshold_Count[ $i$ ]  $\geq$  Sparsity_Threshold_Duration then
14:        sparsity_list[ $i$ ]  $\leftarrow$  True
15:        coeff[ $i$ ].requires_grad  $\leftarrow$  False
16:      end if
17:    else
18:      Below_Threshold_Count[ $i$ ]  $\leftarrow$  0
19:    end if
20:  end for
21: end procedure

```

5.2.2 Forward workflow

This section provides a detailed discussion of the forward workflow. The forward workflow solves the forward problem utilising the predicted BCs $g(\mathbf{x}, t)$, and the known PDE. Although one can use conventional numerical techniques such as finite elements, we decided to use the baseline PINN to solve the forward problem, as it provides a unified workflow. More specifically, in the forward workflow, the inputs and the outputs remain the same with slight modification to the loss terms. To understand the working of the forward workflow, consider a well-posed forward problem as follows:

$$\begin{aligned} \mathbf{u}_t + \mathcal{N}_x[\mathbf{u}] &= 0, \quad \mathbf{x} \in \Omega, t \in [0, T] \\ \mathbf{u}(\mathbf{x}, 0) &= g(\mathbf{x}, 0), \quad \mathbf{x} \in \partial\Omega \\ \mathbf{u}(\mathbf{x}, t) &= g(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega, t \in [0, T] \end{aligned} \quad (5.12)$$

where $\mathcal{N}_x[\mathbf{u}]$ is a differential operator, \mathbf{x} and t are the independent variables of the PDE, Ω and $\partial\Omega$ denotes the spatial domain and the boundary of the problem. The predicted BCs, $g(\mathbf{x}, t)$, from the inverse workflow is also given.

Similar to the inverse workflow, the forward problem can be constrained to satisfy the PDE, the predicted BCs and ICs in terms of mean squared error in Equations 5.13, 5.14 and 5.15.

$$\mathcal{L}_{PDE} = \frac{1}{N_r} \sum_{i=1}^{N_r} |\hat{\mathbf{u}}_t(\mathbf{x}_i, t_i) + \mathcal{N}_x[\hat{\mathbf{u}}(\mathbf{x}_i, t_i)]|^2 \quad (5.13)$$

$$\mathcal{L}_{BC} = \frac{1}{N_b} \sum_{i=1}^{N_b} |\hat{\mathbf{u}}(\mathbf{x}_i, t_i) - g(\mathbf{x}_i, t_i)|^2 \quad (5.14)$$

$$\mathcal{L}_{IC} = \frac{1}{N_0} \sum_{i=1}^{N_0} |\hat{\mathbf{u}}(\mathbf{x}_i, 0) - g(\mathbf{x}_i, 0)|^2 \quad (5.15)$$

$$\mathcal{L}_{forward} = \lambda_{PDE} \mathcal{L}_{PDE} + \lambda_{BC} \mathcal{L}_{BC} + \lambda_{IC} \mathcal{L}_{IC} \quad (5.16)$$

where N_b and N_0 are the number of BCs and IC points. The coefficients λ_{PDE} , λ_{BC} and λ_{IC} in Equation 5.16, helps in balancing the contribution of \mathcal{L}_{PDE} , \mathcal{L}_{BC} and \mathcal{L}_{IC} in the forward workflow. In Equation 5.16, $\mathcal{L}_{forward}$ is the total loss of the forward workflow. The output of the forward workflow provides the reconstructed solution. Although the formulation has been presented for transient problems, the current work focuses solely on steady-state problems.

5.3 Results and discussion

Throughout this work, a FCNN architecture with a scaled exponential linear units (SELU) activation function has been used [163]. The SELU activation function (see Equation 5.17) has self-normalising properties, which facilitates fast and stable training [90].

$$\text{SELU}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha \cdot (e^x - 1) & \text{if } x \leq 0 \end{cases} \quad (5.17)$$

where $\lambda = 1.05$ and $\alpha = 1.67$.

To optimize the network, the Adam optimiser [51] was employed with its default parameters. It is a widely used first order optimiser with adaptive learning rate. The reconstruction error is reported using both the relative L2 error and the maximum absolute pointwise error as follows:

$$\text{Relative L2 error} = \frac{\|\hat{u} - u\|_2}{\|u\|_2} \quad (5.18)$$

$$\text{Maximum absolute pointwise error} = \max_i^N |\hat{u}_i - u_i| \quad (5.19)$$

where $N = N_r + N_b + N_0$ is the total number of points in the training dataset. The sparse measurements were obtained by sampling from FEM (surrogate for the physical system) solution, incorporating 1% and 5% white noise into the temperature distribution. These samples were acquired using Latin-Hypercube sampling Viana [190], applied distinctly to boundary regions and the interior of the domain.

The workflow was implemented on the AccelerateAI nodes, part of the Sunbird high-performance computing (HPC) cluster at Swansea University. The test cases were trained on 2 x AMD EPYC Rome 7452 32-cores and a NVIDIA A100 with 40GB of CUDA memory. It was observed that the test cases did not consume more than 5 GB of CUDA memory.

The performance of the proposed workflow is presented across three test cases. First, the temperature field was reconstructed in two distinct 2D steady-state heat conduction problems, each with unique boundary conditions (BCs), as well as in a 2D problem governed by the Helmholtz equation with a forcing term. Each problem was tested with varying numbers of sensors, and the results are presented using the minimum number of sensors required to achieve an accurate reconstructed solution. Table 5.1 summarises the relative L2 error and the maximum absolute error for each problem with different level of white noise.

Table 5.1: The relative L2 error (%) and the maximum absolute error for each problem with different levels of white noise. For every problem, the first row denotes the relative L2 error, and the second row indicates the maximum absolute error.

Problem	No. of sensors		White Noise		
	Boundary	Interior	0%	1%	5%
1	4	2	1.89	2.46	1.89
			0.04	0.06	0.04
2	4	2	3.15	3.88	8.78
			0.07	0.09	0.11
3	16	10	6.17	3.21	27.3
			0.08	0.11	0.45
3(TL)	4	0	1.22	3.45	22.13
			0.02	0.07	0.31

* TL stands for transfer learning

5.3.1 Problem 1 : 2D steady-state heat conduction with Neumann BC

The first test case is a problem governed by the 2D steady-state heat conduction over a square domain with Dirichlet BC on the left boundary and Neumann BC on the right boundary. Figure 5.3 shows the FEM solution and the sparse measurement points for Problem 1, i.e.,

$$\begin{aligned} \nabla^2 u(x, y) &= 0, & x, y &\in (0, 1) \\ u(0, y) &= 0, & \frac{\partial u}{\partial x} \Big|_{x=1} &= 1 \end{aligned} \quad (5.20)$$

$$\frac{\partial u}{\partial y} \Big|_{y=0} = 0, \quad \frac{\partial u}{\partial y} \Big|_{y=1} = 0, \quad (5.21)$$

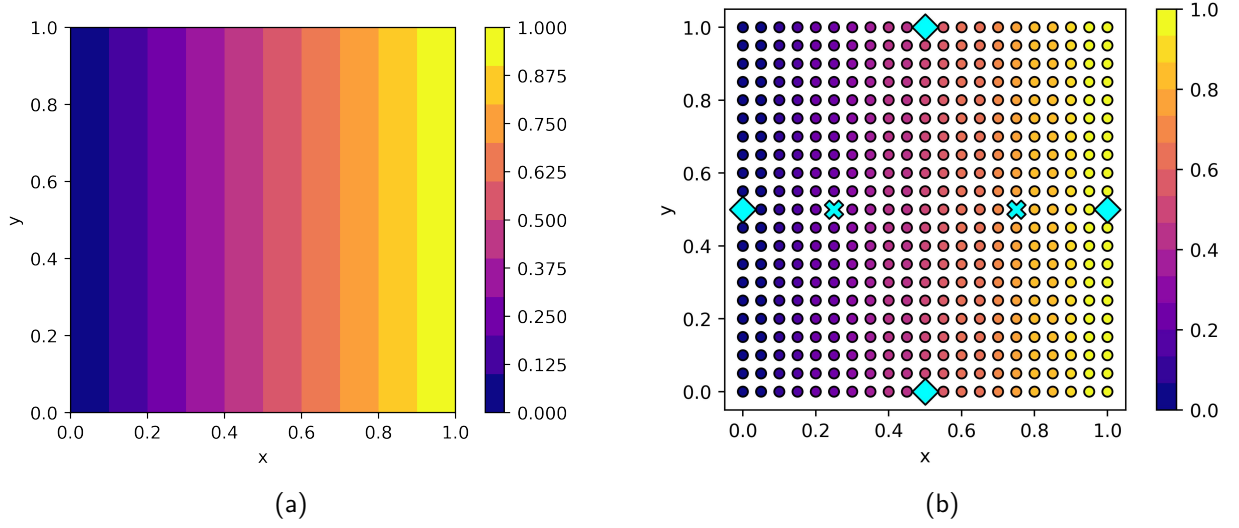


Figure 5.3: The FEM solution (ground truth) of Problem 1 showing the field variable $u(x,y)$ on the left. The location of 4 boundary and 2 interior sparse measurements and the point cloud used for the inverse workflow are shown on the right.

The inverse workflow was trained using a second-order polynomial function $\mathcal{G}(x) = a_0 + a_1x + a_2x^2$, imposed only on the left and right boundaries, for 80k iterations. Figure 5.4, shows the variation of the coefficients of the $\mathcal{G}(x)$ on each boundary with the training iterations.

The sparsity check (see Algorithm 5.1) was initiated after 60k epoch, allowing the NN adequate time to learn the coefficients in the absence of the sparsity check. At the end of the training, very accurate BCs were obtained on the left and right boundaries, specifically yielding precise values for a_0 , a_1 and a_2 . The top and bottom boundaries were unspecified, thus there was no polynomial function associated with these boundaries.

Typically, one would anticipate the errors to increase monotonically with higher noise levels. This is consistent with the error metrics at 0% and 1% white noise. However, the error metrics at 1% white noise were greater than those at 5% white noise (see Table 5.1). This reduction in error can be attributed to overfitting at 1% white noise, where the optimiser may have converged to a local minima but still very close to the global minima.

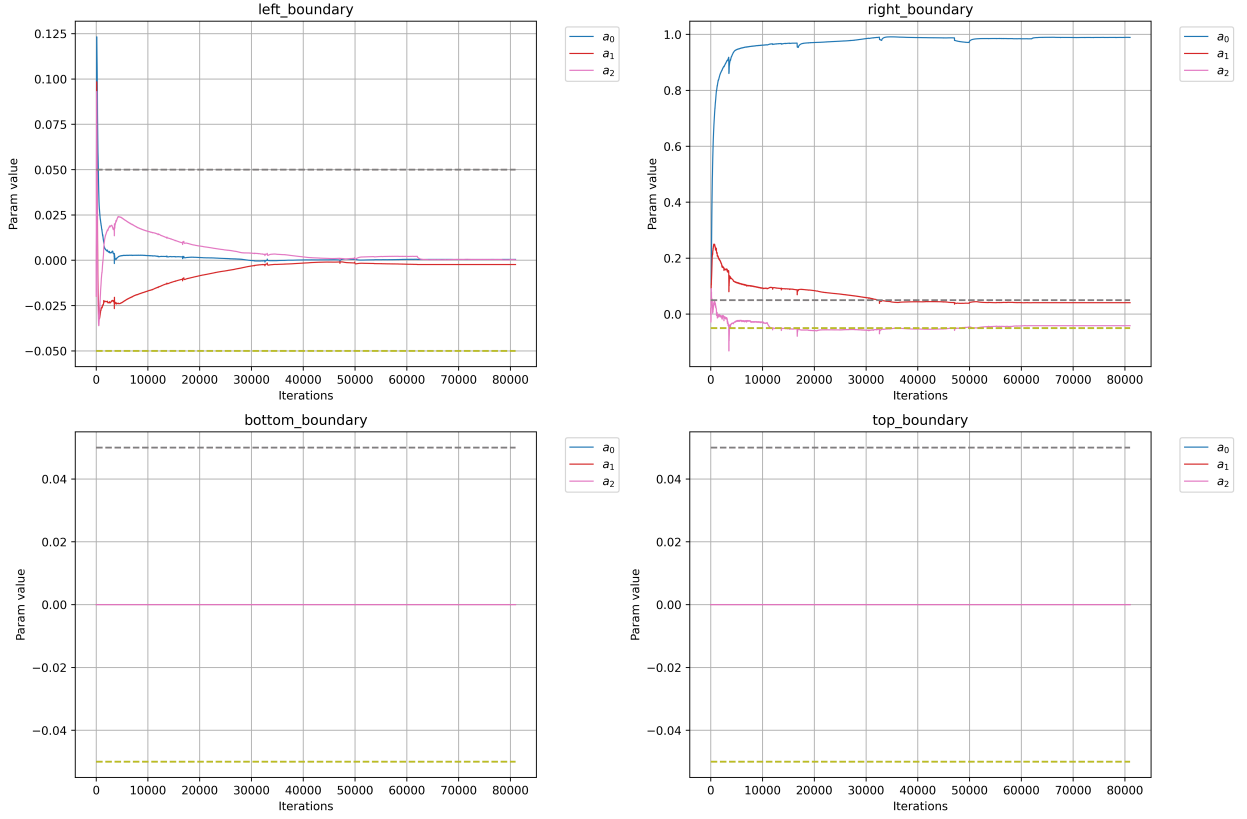


Figure 5.4: The variation of the coefficients of Problem 1 with training iterations. The sparsity knob is shown in dashed grey colour. Any coefficient which remains below the sparsity knob for 2000 consecutive iterations was excluded from further training.

5.3.2 Problem 2 : 2D steady-state heat conduction with a Sine BC

In Problem 2, the Dirichlet BCs are sine functions along all the edges as follows:

$$\begin{aligned} \nabla^2 u(x, y) &= 0, \quad x, y \in (0, \pi) \\ u[x, 0] &= u[x, \pi] = \sin(x), \quad u[0, y] = u[\pi, y] = \sin(y) \end{aligned} \quad (5.22)$$

Figure 5.5 shows the FEM solution and the sparse measurement points for Problem 2. Initially we couldn't converge the training loss of the inverse workflow while using a 2nd order $\mathcal{G}(x)$. To accurately represent the sine BCs, we need at least 5th order $\mathcal{G}(x)$ based on the Maclaurin expansion of sine function in Equation 5.23.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots \quad (5.23)$$

Thus, we trained the inverse workflow with a 7th order $\mathcal{G}(x)$ for 100k iterations. Similar to Problem 1, the sparsity check was initiated after 80k iterations.

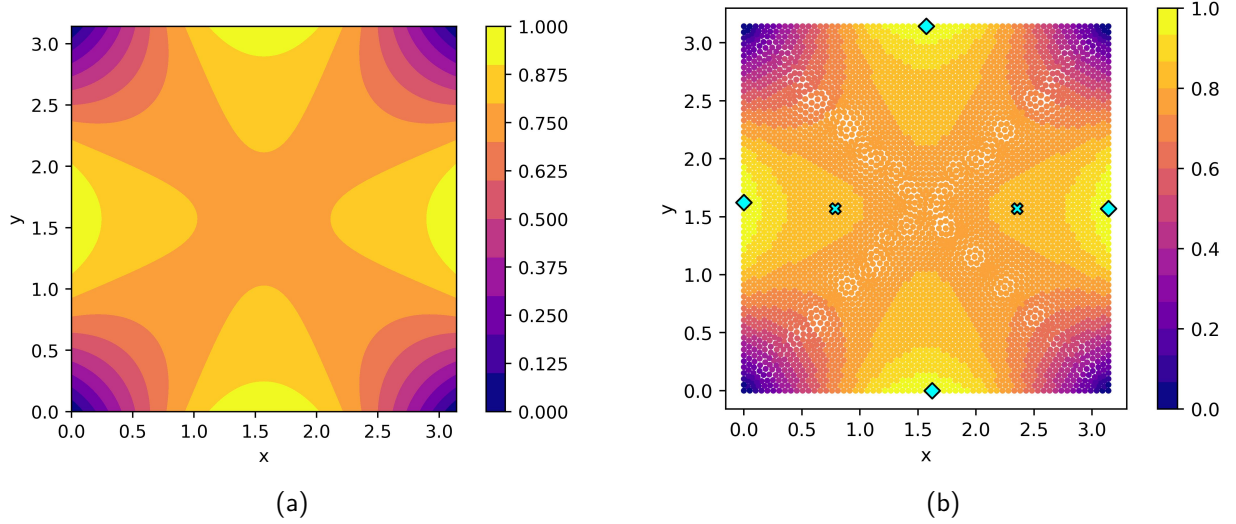


Figure 5.5: The left side of the figure presents the FEM solution (ground truth) for Problem 2 showing the field variable $u(x, y)$. The location of 4 boundary and 2 interior sparse measurements and the point cloud used for the inverse workflow are shown on the right.

Obtaining accurate BCs is particularly difficult in Problem 2 due to the factorial term in the denominator within the Maclaurin expansion of the sine function (Equation 5.23). As the order of the expansion increases, the factorial terms in the denominator grow very rapidly, resulting in smaller coefficients incorrectly passing through the sparsity check, leading to inaccurate BCs. To circumvent this issue, we introduced more training iterations between each successive sparsity check.

This extended period between each successive sparsity check ensures that when a coefficient's magnitude consistently falls below the sparsity threshold, its negligible contribution can be confidently confirmed.

With this extended period between each successive sparsity check, a sensitivity analysis of the Problem 2 with respect to white noise was conducted.. Figure 5.6 illustrates how the relative L2 error does not seem to change over varying levels of white noise across different orders of $\mathcal{G}(x)$, showcasing the workflow's robustness even under noise perturbations. It is critical to highlight that the parameters for the sparsity check require problem-specific tuning to minimise the risk of prematurely discarding the coefficient.

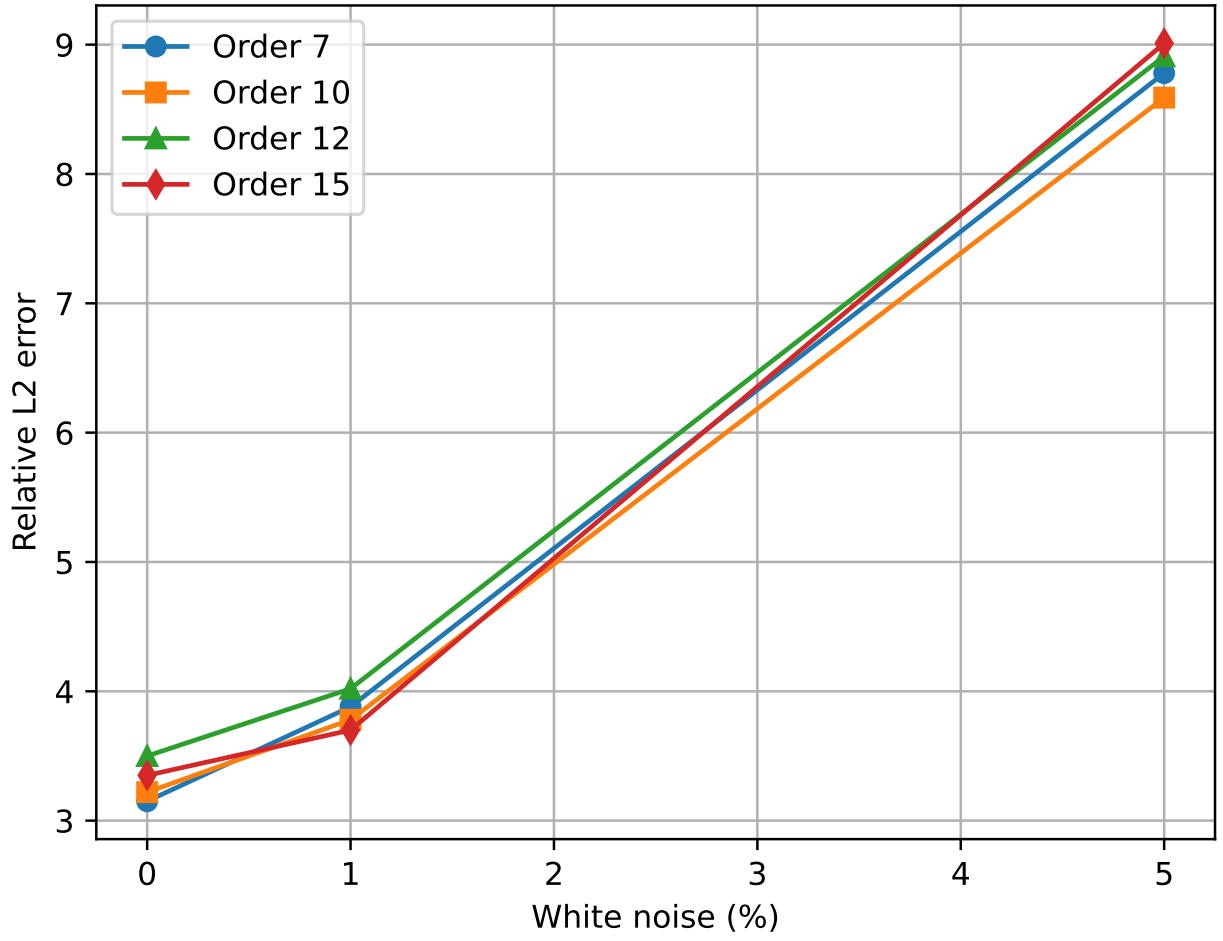


Figure 5.6: Sensitivity of the relative L2 error to white noise across different orders of $\mathcal{G}(x)$ in Problem 2.

5.3.3 Problem 3 : 2D Helmholtz problem over a square domain

The Helmholtz problem is similar to the wave equation, it has wide range of applications in various fields of physics including electromagnetic radiation, seismology, and acoustics. In this section, the 2D Helmholtz solution was reconstructed within a square domain. The equations of the Helmholtz problem are as follows:

$$\begin{aligned}
 \nabla^2 u(x, y) + k_0^2 u(x, y) + f &= 0, & x, y &\in (0, 1) \\
 u(0, y) = u(1, y) = u(x, 0) &= u(x, 1) = 0 \\
 f(x, y) &= k_0^2 \sin(k_0 x) \sin(k_0 y), & k_0 &= 4\pi
 \end{aligned} \tag{5.24}$$

Although the problem is complex, it fortunately has a well-defined analytical solution as outlined in Equation 5.25. Figure 5.7 shows the FEM solution and the sparse measurements for Problem 3. The inverse workflow was trained using a second-order $\mathcal{G}(x)$ for 120k iterations, with the sparsity check initiated after 80k iterations.

$$u(x, y) = \sin(k_0 x) \sin(k_0 y) \quad (5.25)$$

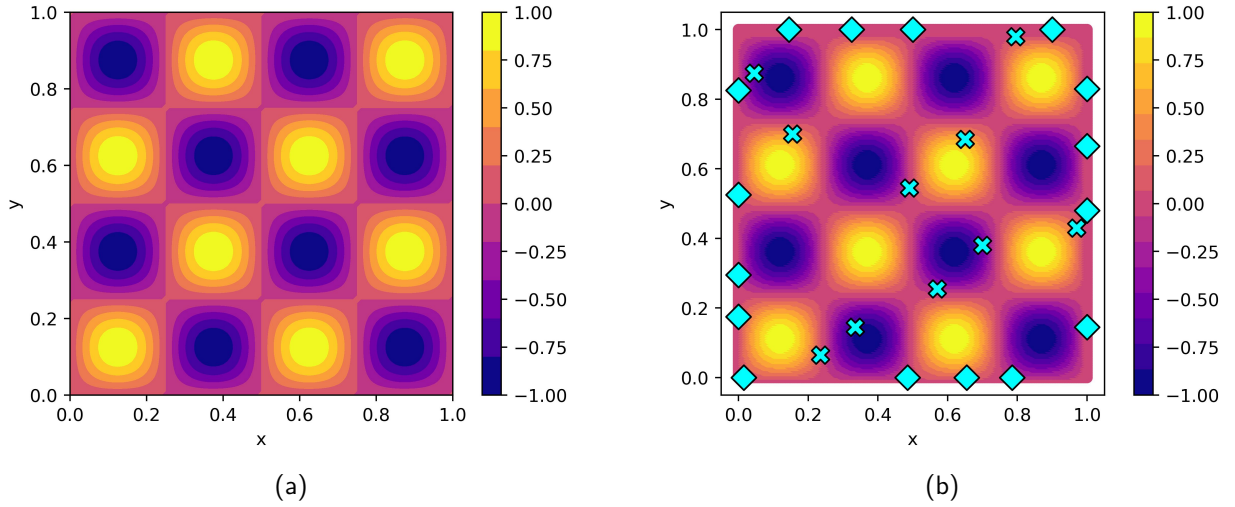


Figure 5.7: The FEM solution (ground truth) of Problem 3 showing the field variable $u(x, y)$ on the left. The location of 16 boundary and 10 interior sparse measurements and the point cloud used for the inverse workflow are shown on the right.

After experimenting with various combinations of boundary and interior sparse measurements, we determined that a minimum of 26 sparse measurements was necessary to successfully reconstruct the solution when training the model from scratch using a 5th order \mathcal{G} . Other combinations resulted in the training loss not converging in 120k iterations.

Transfer learning was employed to accelerate convergence and reduce the number of sparse measurements. While transfer learning is not mandatory, it offers a number of benefits compared to training the inverse workflow from scratch. This approach involves leveraging knowledge from a similar problem to reconstruct the original problem.

A problem with $u(x, y) = 0.15$ as the BC on all edges was randomly selected. The FEM solution for this problem was used to train the inverse workflow. This initial training phase provided us with pre-learned parameters such as network weights, and the coefficients of \mathcal{G} . These pre-learned parameters are then utilised to train the inverse workflow for our original problem. The entire process is summarised in Figure 5.8.

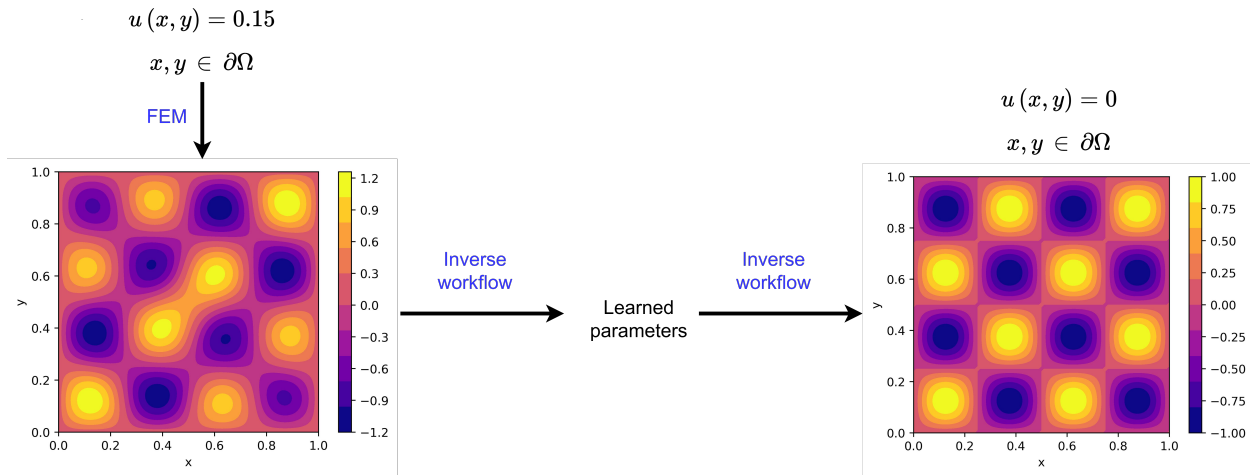


Figure 5.8: A flow chart of the transfer learning approach used in Problem 3..

With the transfer learning approach, the reconstruction only required 4 sparse measurements on the boundary with a significant decrease in the training time. For instance, the training loss of the inverse workflow converged in just 14k iterations, a significant reduction from the previous 120k iterations.

As depicted in Table 5.1, the relative L2 error exhibits a modest increase when transitioning from 0% to 1% white noise, however, a dramatic increase is observed from 1% to 5% noise levels, highlighting a threshold beyond which the model's sensitivity to noise significantly impacts its accuracy.

5.4 Conclusion

This chapter discussed a PINN-based workflow for reconstructing the solutions of physical systems governed by PDEs, utilising sparse measurements and a foundational understanding of the inverse problem. Traditional PINN based methods, along with other approaches for solution reconstruction, have historically relied on combining sparse experimental data with either extensive number of experiments, or the deployment of an impractically high number of sensors in an experiment, facing significant practical challenges due to the frequent unavailability of large datasets or the explicit knowledge of BCs.

Unlike these methods, the proposed workflow only requires the knowledge of the governing PDE and an approximate range of the BCs for the solution reconstruction, thus, reducing the dependency on large volumes of experimental or simulation data. The proposed workflow is a suitable choice for solution reconstruction in cases where acquiring a large quantity of experimental

data is difficult, instead expertise in the subject matter can be leveraged to obtain the PDE and the approximate range of the BCs.

In all test cases, sparse measurements were obtained through Latin-hypercube sampling. Yet, optimising sensor placement could further enhance the accuracy of the solution reconstruction. The proposed workflow was tested against two distinct 2D steady-state heat conduction problems, each characterised by unique BCs, as well as a complex 2D problem governed by the Helmholtz equation. Table 5.1, summarises the relative L2 error for each problem with different levels of white noise. Each problem was tested with different configurations of sensors, and the values listed in Table 5.1 represent the minimum number of sensors required to obtain an accurately reconstructed solution.

The proposed workflow has delivered promising results, eliminating the need for a pre-trained data-driven model, that typically requires a large amount of training data. It was observed that transfer learning, when a solution to a similar problem is available, not only accelerates the convergence of the training loss but also reduces the number of sparse measurements required.

Our approach contributes to the ongoing effort to address the challenges of inverse problems by leveraging minimal information to convert ill-posed scenarios into more manageable, approximately well-posed problems. Theoretically, the workflow is expected to be effective for most inverse problems involving PDEs, provided there are sufficient sparse measurements and the aforementioned prior knowledge is known. However, PINNs are known to have issues in solving forward and inverse problems with discontinuities. Thus, the proposed workflow may face issues solving inverse problems with discontinuities. Future work could focus on integrating sensors that account for uncertainties in their placement, and on transient problems.

Chapter 6

HIVE digital twin

Inverse problems, often ill-posed with limited data, become complex with non-linearities in partial differential equations (PDEs) and noisy data. Traditional brute-force search methods are inefficient. Baseline PINN offers a straightforward solution but typically requires many sensors or explicit knowledge of the BCs, limiting their practicality in inverse problems. As discussed in Chapter 5, unlike traditional numerical methods for PDEs, PINNs blend data-driven and physics-based techniques. This hybrid approach ensures that the IC, BCs, and the governing PDE are inherently satisfied, allowing effective resolution of forward problems. For inverse problems, where some elements may be unknown or ambiguous, PINNs are capable of learning the missing information from sparse data.

Fusion experiments are inherently challenging and expensive, making the generation of thousands of experimental data for conventional machine learning based solution reconstruction impractical. Additionally, the BCs in such experiments are often elusive, preventing straightforward forward simulations using conventional numerical techniques to generate synthetic data. This chapter focuses on the reconstruction of the temperature field using sparse measurements, as well as the construction of a foundational physics-informed digital twin based on the HIVE experiment.

6.1 HIVE experiment

Heat by Induction to Verify Extremes (HIVE) is an experimental facility at the UK Atomic Energy Authority (UKAEA) to expose the plasma facing components (PFC) to the high thermal loads that they will experience in a fusion reactor. It was established back in 2015 to enable early verification of the thermo-fluid performance of PFCs. Instead of full component qualification, HIVE focuses on the comparing different designs and manufacturing techniques [191]. The primary goals of

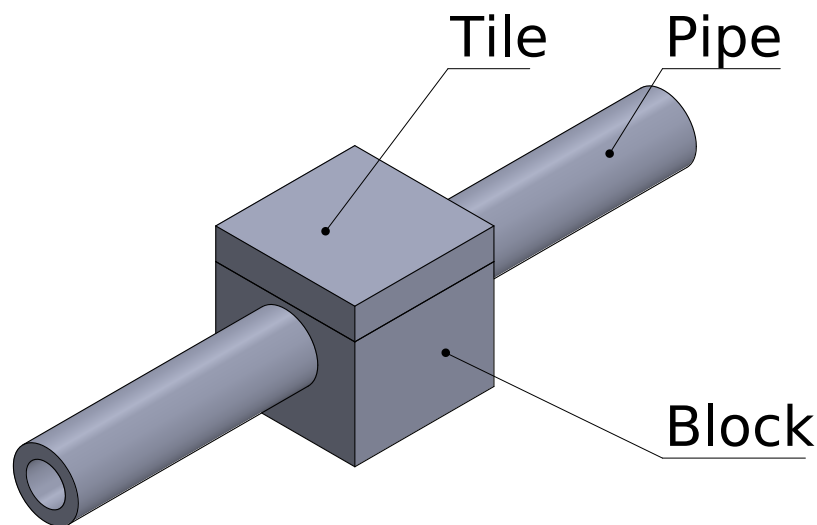


Figure 6.1: A generic HIVE sample showing its individual components.

HIVE are:

- Analyse the thermal performance of a component.
- Investigate concepts which employ novel features or materials.
- Provide a comparison between different manufacturing methods.
- Validate computational models.

A schematic of the HIVE experimental setup, including the vacuum vessel, ports, and cooling system, is provided in Chapter 1 (Figure 1.2). This setup highlights the critical elements used in testing PFCs, such as the induction coil and infrared camera, which facilitate accurate thermal loading and monitoring. Figure 6.1 shows a generic HIVE sample, highlighting its various components. The tile represents the plasma-facing side, while the pipe is designed for coolant flow to cool the component. To perform an experiment, the component is fitted in a mounting bracket. A coil design is chosen and is positioned relative to the component. During an experimental campaign, the component is thermally loaded by induction heating whilst being actively cooled with pressurised water [8].

6.1.1 Induction heating

Induction heating is a process of heating electrically conductive materials (usually metals) by electromagnetic induction. An alternating current (AC) passes through an induction coil, creating

a rapidly alternating magnetic field around the coil. When a conductive material is placed within this alternating magnetic field, the field induces eddy currents within the material. The material's electrical resistance converts the energy from the eddy currents into heat. This heating occurs primarily on the surface of the material, with the intensity of heating diminishing with depth (a phenomenon known as the skin effect). The induction heating system used by HIVE has a frequency range of 50 kHz to 150 kHz, and usually operates around 100 kHz.

6.1.2 Active cooling

Cooling for HIVE is managed by a closed-loop temperature control unit, which sets the temperature, pressure, and flow rate of the coolant. The system supports a flow rate of 51 to 801/min, pressure between 0.4 and 2 MPa, and temperatures from 25°C to 200°C, with higher temperatures achievable only at high pressures.

6.2 The AMAZE sample

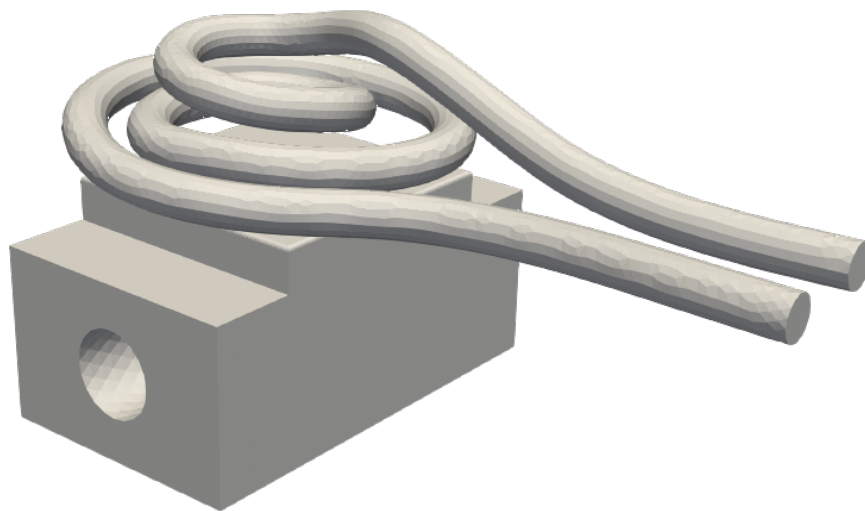


Figure 6.2: The AMAZE sample with the coil setup.

The sample used in this work was manufactured as part of AMAZE (additive manufacturing aiming towards zero waste and efficient production of high-tech metal products), an European FP7 project aiming to grow confidence in additive manufacturing [192]. A detailed discussion of the HIVE experimental setup and testing of the AMAZE sample can be found in Hancock *et al.* [193]. In this thesis, the sample will be referred to as the AMAZE sample.

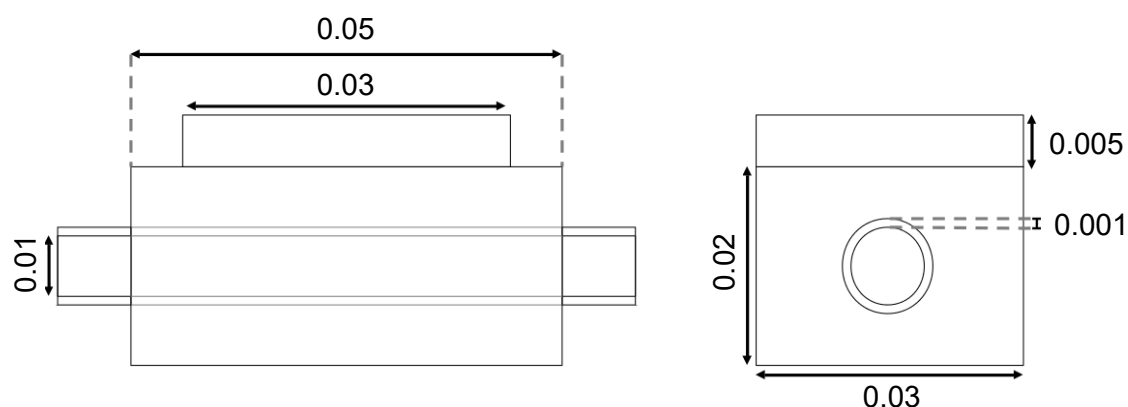


Figure 6.3: Drawing of the AMAZE sample. All dimensions are shown in meters.

Figure 6.2 shows the AMAZE sample, featuring the coil arrangement relative to the sample. The sample is a copper block on a copper pipe with a tungsten tile on the top. Figure 6.3 shows the dimensions of the AMAZE sample.

6.3 Numerical simulation with VirtualLab

VirtualLab [194] is a modular platform that enables users to run simulations of physical laboratory experiments, essentially serving as their 'virtual counterparts.' It functions as a streamlined wrapper for Apptainer images to carry out desired tasks. Apptainer (formerly Singularity) is a container technology designed to sandbox applications without the overhead of a full-blown virtual machine, instead utilising the host's Linux kernel [195]. VirtualLab supports Apptainer images for open-source programs such as Salome [196], Cad2Vox [197], PyTorch, ParaVis [198] etc. The core idea of VirtualLab is to facilitate easy parameterisation, parallelisation, and portability [199]. The author of this thesis contributed to testing and developing bash scripts to run VirtualLab on GPU-enabled nodes on the Sunbird HPC cluster.

Lewis [200] conducted extensive simulations of the AMAZE sample in VirtualLab. The simulation was divided into three parts: induction heating of the coil, fluid flow in the pipe and thermal simulation of the sample. The coil emits a volumetric heat load based on the input current and its location relative to the AMAZE sample. The electromagnetic analysis of the coil was carried out using an open-source library called Electric Regularized Maxwell Equations with Singularities (ERMES) [201]. Once the electric field \mathbf{E} induced within each element by the changing magnetic field due to induction heating and eddy current density \mathbf{J} within each element

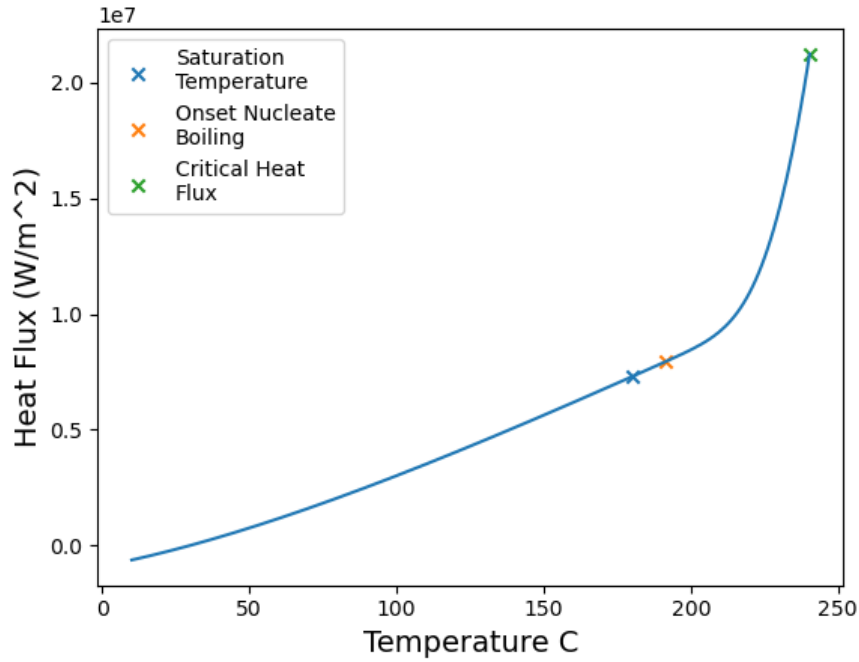


Figure 6.4: The 1D boiling curve used in HIVE. The curve illustrates different boiling regimes, marked by key points: saturation temperature (blue 'x'), onset of nucleate boiling (orange 'x'), and critical heat flux (green 'x'). Heat flux increases non-linearly with temperature, peaking at the critical heat flux before transitioning to film boiling.

are solved, the volumetric heat source (\dot{Q}) can be calculated for each element using:

$$\dot{Q} = \mathbf{J} \cdot \mathbf{E} = \sigma |\mathbf{E}|^2 \quad (W/m^3) \quad (6.1)$$

where σ is the conductivity of the material for each element. The coolant flowing through the pipe was modelled as 1D pipe flow, defined using a boiling curve that relates the heat flux (q) between the coolant and the pipe as a function of the pipe wall temperature (T_{wall}). As the value of the T_{wall} increases, the curve traverses four regimes: natural convection, nucleate boiling, transition boiling, and film boiling. Ideally, the coolant should remain in the nucleate boiling regime due to its high heat transfer efficiency [202]. This relationship defines a Robin BC, where the heat flux depends on T_{wall} as shown in Figure 6.4. Finally, the volumetric heat load and heat flux from the boiling curve were applied as BCs in the thermal simulation of the sample.

To conduct the numerical simulation, several tools within the VirtualLab platform were utilised. Salome was used for both pre-processing and post-processing, with geometry creation and meshing performed in Salome. For post-processing, Salome employs a modified version of ParaView called ParaVis, which supports the MED meshes, a file format preferred by Salome. The finite element analysis was carried out using Code_Aster [203], an open-source solver with over 40 years of

development. All these components are integrated into the Salome-Meca GUI [204], providing a streamlined and user-friendly interface for the entire simulation workflow. Additionally, ERMES was integrated into the Salome-Meca workflow through offscreen rendering, as the standard GUI does not support third-party solvers directly.

In summary, the coil serves as a volumetric heat source term (W/m^3), and the coolant acts as a Robin BC (W/m^2) for the AMAZE sample. This configuration reduces a multi-physics problem involving electromagnetism and fluid flow to a heat transfer problem. The steady-state behaviour of the components tested in HIVE is of interest, as these are the conditions they will be subjected to for long periods in a fusion device. Since there is no Dirichlet BC, the initial temperature of the coolant (30°C) was used as the initial condition for the solver to speed up convergence. This allows the solver to iteratively compute the temperature distribution until it reaches a steady state, accurately reflecting long-term operational conditions. The thermal analysis was conducted using the MUMPS, a memory efficient sparse solver [205].

6.4 Physics-informed digital twin

This section details the foundational physics-informed digital twin developed using the PINN-based workflow discussed in Chapter 5. The PINN-based workflow which does not rely on additional experimental data, is particularly suitable for situations where data acquisition is challenging.

6.4.1 Assumptions

Given the complexity of the HIVE simulation, several simplifications were implemented in the Lewis's FEM simulations to manage this complexity effectively.

- The AMAZE sample material is homogeneous and composed entirely of copper. Although this is not a reasonable assumption, PINNs struggle with forward problems involving heterogeneous materials, making it impractical to use them for inverse modelling in such cases. Therefore, to simplify the initial setup, homogeneous material properties are assumed, resulting in perfectly bonded contacts between the tile-block and tile-pipe. The assumption of homogeneity limits the robustness of the workflow by simplifying the physical model and excluding material property variations, such as spatially varying thermal conductivity. PINNs currently struggle with forward problems involving discontinuities, making it challenging to extend the workflow to heterogeneous materials. Addressing this would require architectural improvements to better handle such complexities.

- The coil placed on top of the tile doesn't transfer significant heat to the bottom of the sample. Therefore, it is assumed that anything below the pipe isn't receiving any heat. Although this reduces the complexity of the model, this assumption significantly decreases the size of the heat source array by almost half, thereby saving time in training the PINN.
- The Robin BC due to the pipe flow is already known beforehand, as it can be approximated based on the boiling curve, which provides the relationship between heat flux and wall temperature, T_{wall} . This assumption aligns with the 1D flow simplification made in Lewis's simulations, where flow boiling in the pipe, typically a complex two-phase flow, was approximated for computational feasibility. This prior knowledge simplifies the inverse problem further since one of the boundary conditions is already known. However, it is important to acknowledge that real-world scenarios could introduce variations and uncertainties.

These simulations served as the ground truth for evaluating the performance of the PINN-based digital twin developed in this chapter. These simulations provided temperature fields under controlled conditions, which were then used to compute the errors and validate the accuracy of the digital twin's reconstructed temperature distributions.

6.4.2 Revisting the PINN-based workflow

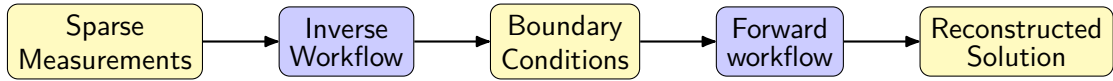


Figure 6.5: The PINN-based workflow for solution reconstruction.

Figure 6.5 shows a high-level overview of the PINN-based workflow, with detailed explanations available in Chapter 5. This workflow reconstructs the solution directly from sparse measurements without requiring extensive training with numerous experiments. Sparse measurements are fed into the inverse workflow to determine the BCs, which the forward workflow then uses to obtain the reconstructed solution by solving a forward problem. The inverse workflow requires knowledge of the PDE (Equation 6.2), the sparse measurements, and approximate bounds of the solution. Optionally, transfer learning can be used accelerate convergence and decrease the number of sparse measurements.

$$\kappa \Delta u = f . \quad (6.2)$$

6.5 Results and discussion

6.5.1 Initial simulations to identify suitable hyperparameters

For the initial simulations, the coolant parameters were set to a temperature of 30°C, a pressure of 1 MPa, and a velocity of 10 m/s. The coil was driven with a current of 300A and a frequency of 10 kHz. Figure 6.6 shows the temperature distribution of the AMAZE sample using the numerical simulations developed by Lewis [200], with simplifications introduced in Section 6.4.1.

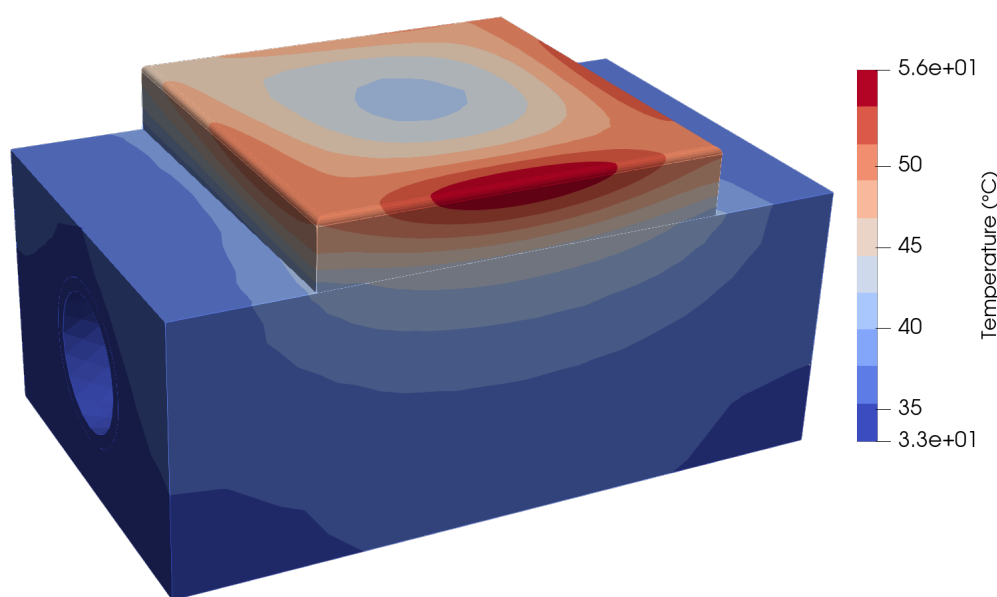


Figure 6.6: The temperature profile from the thermo-mechanical simulation of the AMAZE sample.

To reconstruct the solution, 50 pseudo-random sparse measurements were sampled using Latin-hypercube sampling. An approximate heat source range was used by expanding the bounds of the actual heat source by 25%. A FCNN with the Adam optimiser was employed to train both the inverse and forward workflow. Several hyperparameters were tested, including network size, activation function, and the order of the polynomial used to fit the BCs.

The results indicated that a 10th-order polynomial was sufficient to accurately capture the complexities of the heat source term. Various activation functions were also tested, including hyperbolic tangent (Tanh), rectified linear unit (ReLU), and sigmoid linear unit (SiLU). The lowest relative L2 error of 3.45% was achieved using the ReLU activation function.

6.5.2 Transfer learning

Lewis [200] complemented his simulations with the development of a data-driven surrogate model using a simple FCNN. This model was designed to reconstruct the solution by taking sparse measurements as input and providing the reconstructed solution as output. His approach utilised data from 1000 experiments, successfully achieving accurate reconstructions with just 3–4 sparse measurements per experiment. In contrast, this work relies solely on 50 sparse measurements from a single experiment without requiring additional experimental data, demonstrating the feasibility of a PINN-based approach in handling such inverse problems with minimal data.

It is important to note that while reconstructing the solution of such a complicated problem with just 50 sparse measurements without any training on numbers of experiments is exceptional, it is not practical. The small size of the sample makes it challenging to fit 50 thermocouples, as there simply isn't enough space to accommodate that many sensors.

Inverse problems are known for having non-unique solutions, and navigating from numerous local minima to the specific local minimum that generates the correct solution is a challenging task. Additionally, the ill-posed nature of inverse problems further complicates this process as the global minimum does not necessarily represent the physical solution being sought. When the weights are randomly initialised, this process becomes even more time-consuming as the model has to spend more iterations finding a suitable solution. In contrast, using transfer learning with pre-trained weights significantly reduces the time required to reach an acceptable solution, as the model starts from a more informed state rather than from scratch.

To circumvent this issue, transfer learning was employed by training the workflow on a simpler problem - not necessarily a low-fidelity solution of the problem at hand, but rather any random heat source term within the approximate range of the heat source and the same PDE. This approach allows the weights to be more informed about the problem they need to tackle. In this method, only one FEM solution is required, avoiding the necessity for multiple experiments typically needed in data-driven models.

As demonstrated in Table 6.1, a linear increase in the current (A) of the coil was simulated, and solutions were reconstructed using both independent steady-states and transfer learning. For transfer learning, a constant heat source of 500 W/m^3 on the tile was initially used to train the model. This constant thermal load was not used in the independent steady-state solutions but served as an initial training step to provide a baseline for the weights in the transfer learning. For subsequent time steps with varying current values, transfer learning was applied using the model trained on the previous time step, allowing the model to progressively adapt to the changing conditions. With this approach, the solution was reconstructed using just 10 sparse measurements,

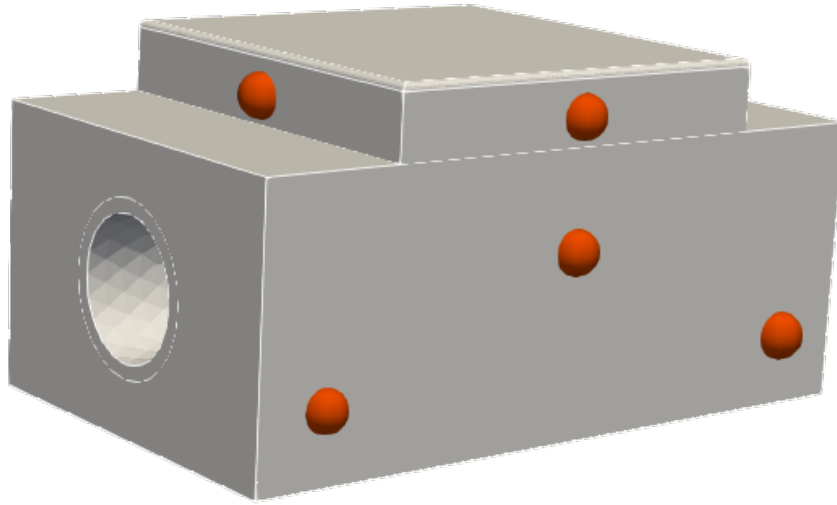


Figure 6.7: Manually sampled 10 sparse measurements are shown on the visible surfaces of the AMAZE sample. The remaining 5 sparse measurements, not visible in this figure, are mirror images on the opposite, invisible side.

as shown in Figure 6.7. This methodology significantly reduced the average training time from 50 minutes to 25 minutes, compared to independently reconstructing each steady-state.

Table 6.1: Comparison of relative L2 errors (in %) and average time taken (min) for different currents (A) using independent steady-states and transfer learning.

Current (A)	50	100	150	200	250	300
Independent steady-states						
Rel L2 error	3.45	5.12	2.98	4.56	3.33	5.77
Average time taken: 50 mins						
Transfer learning						
Rel L2 error	3.45	5.88	4.21	3.09	5.34	2.47
Average time taken: 25 mins						

The reconstructed temperature field can be used to assess whether the material's maximum temperature limit has been exceeded. Additionally, if the relationship is known, parameters like temperature-dependent conductivity can also be predicted. In the case of Copper the thermal conductivity barely changes after 100K and nothing interesting occurs [206]. For demonstration purposes, a hypothetical relationship between thermal conductivity and temperature is considered to illustrate how a digital twin might predict material properties reaching critical values. The initial conductivity $\kappa_{initial}$ of copper at 30°C is $394 \text{ Wm}^{-1}\text{K}^{-1}$. Let us assume a hypothetical relationship between thermal conductivity and temperature as follows:

Table 6.2: The variation in the minimum thermal conductivity minimum with increasing input current, calculated using the hypothetical relationship between thermal conductivity and temperature.

Current (A)	50	100	150	200	250	300
κ_{min}	394	387.39	381.34	377.72	373.39	370.2

$$\kappa(X, Y, Z) = \kappa_{initial} - 0.0001(T(X, Y, Z) + T(X, Y, Z)^2 + T(X, Y, Z)^3) \quad (6.3)$$

Table 6.2 demonstrates the variation in the minimum thermal conductivity (κ_{min}) of copper as the input current increases from 50 A to 300 A. The table shows a decrease in κ_{min} with increasing current, indicating how the hypothetical relationship between thermal conductivity and temperature affects the material properties under different heat loads.

6.6 Conclusion

The PINN-based workflow demonstrated the capability to accurately reconstruct field variables from sparse measurements, significantly reducing the need for extensive experimental data. The workflow requires knowledge of the PDE, the sparse measurements, and the approximate bounds of the solution to be reconstructed. The transfer learning approach, proved effective in minimising reconstruction time and error. This methodology shows promise for efficient and practical applications in data-scarce environments.

This methodology has been integrated as a foundational physics-informed digital twin, exploring the possibilities of building a comprehensive digital twin for fusion energy applications. While still in its early stages, involving a number of assumptions, this approach allows the reconstructed solution to predict parameters such as peak temperature and temperature-dependent conductivity. This capability facilitates proactive adjustments, such as stopping the experiment if the parameter approaches a critical value. Thus, the digital twin enhances the utility and efficiency of monitoring and controlling experimental conditions.

However, the digital twin presented in this chapter has notable limitations. The simulations were conducted at relatively low temperatures, with a maximum temperature of 56°C, which does not fully capture the extreme thermal environments typical of fusion reactors, where plasma-facing components experience much higher temperatures. This limitation was intentional, as focusing on lower temperatures allowed for the simplification of BCs (e.g., neglecting radiation heat transfer) and reduced problem complexity. These simplifications were critical for the initial development

and validation of the digital twin framework.

Future developments will need to address the challenges posed by higher temperatures, including the incorporation of radiation BC, temperature-dependent material property variations, and non-linear effects caused by large thermal gradients. While PINNs have shown potential, their application to these more complex scenarios may require enhanced architectures and advanced training strategies to capture the sharp gradients and non-linearities associated with extreme conditions.

Chapter 7

Conclusion and future work

7.1 Conclusions

In this thesis, the primary aim was to investigate the suitability of PINNs for solving inverse problems and to extend their application in constructing a foundational physics-informed digital twin for replicating physical experiments. This research was driven by the challenges inherent in fusion reactor environments, where components must withstand extreme conditions that make data collection difficult and often result in limited or noisy measurements. The complexity of these environments and the limited availability of diagnostic tools necessitate alternative techniques for inferring unknown BC, material properties and field variables from sparse data.

PINNs were chosen for their potential to address the challenges posed by scenarios where conventional data collection is challenging or where simulation data is difficult to generate due to unknown or elusive BCs. Despite their advantages, the novelty of PINNs and the potential quirks in their application required a detailed study to fully understand their strengths and weaknesses. The research involved solving thermal problems, including those with discontinuities in the solution or input parameters, and parametric problems that required hyperparameter tuning. A number of pros and cons of PINNs were identified and are discussed in further sections.

7.1.1 Advantages of PINNs

Reducing dependence on experimental or simulation data

Conventional deep learning-based PDE solvers typically rely on large amounts of experimental and simulation data. However, PINNs can operate effectively even in the absence of such data, and they can also easily integrate noisy and sparse experimental or simulation data into the framework

when solving inverse problems. This makes PINNs a unique candidate for solving problem involving PDEs, especially in scenarios where conventional numerical and deep learning approaches are impractical, as discussed in Chapter 2.

Integrating the governing differential equations

Conventional deep learning methods do not directly incorporate the governing differential equations into their frameworks. However, these equations can provide crucial information during the training process, reducing the reliance on labelled data alone. By embedding the governing differential equations, PINNs are less prone to overfitting, particularly in forward problems, as they aim to solve a well-posed problem rather than merely fitting the NN to match the input and output data.

Mesh independence

Meshing is a repetitive and time-consuming task in traditional numerical methods. PINNs performs the optimisation on point clouds rather than relying on a mesh, allowing them to efficiently handle complex geometries and irregular domains. Even with domain decomposition, as discussed in Chapter 2, the number of decomposed parts is far fewer than the number of elements that would typically be used in the same domain. For example, in Chapter 2, the lid-driven cavity problem was solved using just three sub-domains with PINNs. In contrast, solving the same problem with traditional methods like FEM or FVM would require a significantly higher number of elements.

Handling high-dimensional parametric problems

PINNs have demonstrated their capability to handle high-dimensional parametric problems effectively, as highlighted in Chapter 2 and 3. Unlike traditional numerical methods that can struggle with the curse of dimensionality, PINNs can efficiently manage multiple input parameters. However, it was observed that output-dependent inputs, such as temperature-dependent conductivity, present significant challenges. In these cases, the loss landscape becomes dynamic, making it difficult for the optimiser to converge to a stable solution.

7.1.2 Disadvantages of PINNs

Addressing discontinuities and hyperparameter sensitivity

PINNs, as neural networks, are inherently continuous functions when continuous activation functions are used. This continuity can make it challenging to solve problems involving discontinuous BCs,

conductivity, or other discontinuous parameters. Various approaches, such as importance sampling, the use of signed distance functions, or careful selection of hyperparameters (as discussed in Chapters 3 and 4), can help mitigate the impact of discontinuities. However, as demonstrated in Chapter 4, using a discontinuous activation function like ReLU can lead to adverse effects, such as vanishing gradients, complicating the training process.

In general, the Tanh activation function with FCNN and learning rate of 10^{-3} is considered a good starting point for training PINNs. However further investigation might be required on a case-by-case basis to address specific problems. To assist with this, a list of optimal hyperparameters has been presented in Chapter 4, aiming to reduce the number of necessary trials during the training process.

Imbalanced loss terms

PINNs involve a multitask loss function where different loss components such as the BC loss, PDE loss, and others compete with each other. A common issue arises because the pointwise PDE losses, which involve derivatives, tend to have very low magnitudes, leading to imbalanced loss terms. This imbalance can cause the NN to prioritise certain loss components over others, potentially hindering the training process. To address this, it often becomes necessary to adjust the coefficients of each loss term to balance their contributions to the overall loss.

7.1.3 Solution reconstruction workflow

A key outcome of this research was the development of a PINN-based workflow for reconstructing field variables, such as temperature, from sparse measurements. This workflow was validated across several test cases and demonstrated its effectiveness in reducing the reliance on large experimental datasets. While the workflow does not require a low-fidelity solution, it produced promising results by leveraging similar but simpler problems for transfer learning. This was exemplified by solving the 2D Helmholtz problem in Chapter 5. The transfer learning approach not only accelerated the convergence of the training loss but also reduced the number of sparse measurements needed.

7.1.4 Digital twinning

The final objective was to construct a fundamental digital twin for the HIVE experiment, using the developed PINN-based workflow. Transfer learning was effective in minimising reconstruction time and error, showcasing efficient applications in data-scarce environments. This digital twin facilitated proactive adjustments, such as stopping the experiment if the parameter approaches

a critical value. This workflow is not confined to fusion experimental facilities, it can also be applied to any experiment where obtaining experimental data is particularly challenging. The shortcomings and potential solutions are discussed in the next section.

7.2 Future work

7.2.1 Better diagnostics

While the PINN-based workflow was successfully tested against simulation data with white noise, future work should focus on applying this workflow to experimental data, which may reveal interesting trends and challenges. However, there are practical limitations to consider, such as the restricted placement of thermocouples, which cannot be placed directly on the tile due to the risk of melting. Predicting the volumetric heat load under these constraints may lead to inaccuracies in solution reconstruction. Therefore, the development and integration of more advanced diagnostic tools would be beneficial to overcome these limitations and improve the accuracy and reliability of the reconstructed solution.

7.2.2 Reconstruction of complicated simulations

Several simplifications were made to reduce the complexity of the actual HIVE simulation with the AMAZE sample. One of these simplifications was assuming the AMAZE sample to be homogeneous, despite it being composed of heterogeneous material with nonlinear properties. Additionally, all reconstructions were performed solely for the thermal simulation, with fluid flow in the pipe and the volumetric heat load from the coil pre-calculated using finite element methods.

The foundational digital twin was demonstrated on a steady-state example, but it can be easily extended to handle temporal problems with minimal effort by simply incorporating time as an additional feature in the training data. Incorporating all these complexities into the foundational digital twin would represent a significant advancement in accurately simulating and reconstructing the behavior of the HIVE and similar facilities.

7.2.3 Faster reconstruction

The time required to reconstruct the solution and provide feedback was more than 20 minutes. While this is a significant achievement, obtaining the reconstructed solution faster, or even in real-time, would be even more beneficial. One approach to achieving this is by using more

accurate cases for transfer learning or employing ensembling techniques, which can enhance the model's accuracy and speed on a case-by-case basis. Additionally, hardware and software upgrades, such as leveraging specialised hardware like GPUs or TPUs, using compiled languages for performance-critical code, or optimising the operating system environment, could further reduce the reconstruction time.

7.2.4 Other machine learning techniques

In addition to PINNs, other machine learning techniques can be explored to enhance the accuracy and reliability of solution reconstruction. Bayesian PINNs, for instance, can be used to incorporate confidence intervals and uncertainty estimates, providing a more robust understanding of the predictions [207]. Additionally, physics-informed DeepONets are alternative approach, where neural operators are trained to approximate the solution operator of PDEs directly, enabling the reconstruction of complex physical systems with limited data [208]. These operators can solve parametric PDEs even more efficiently than traditional PINNs, making them particularly valuable in scenarios where capturing uncertainty and complex relationships is critical.

Bibliography

- [1] O. Wallach. "Race to net zero: Carbon neutral goals by country". Accessed: 2024-06-03. (Jun. 2021), [Online]. Available: <https://www.visualcapitalist.com/sp/race-to-net-zero-carbon-neutral-goals-by-country/>.
- [2] H. Ritchie, M. Roser and P. Rosado. "Energy". Accessed: 2024-06-03. (Oct. 2022), [Online]. Available: <https://ourworldindata.org/energy-production-consumption>.
- [3] Department for Energy Security and Net Zero, *Towards fusion energy 2023: The next stage of the uk's fusion energy strategy*, Online, Accessed: April 20, 2024, 2023. [Online]. Available: <https://assets.publishing.service.gov.uk/media/65301b78d06662000d1b7d0f/towards-fusion-energy-strategy-2023-update.pdf>.
- [4] S. Ciattaglia, M. C. Falvo, A. Lampasi and M. Proietti Cosimi, "Energy analysis for the connection of the nuclear reactor demo to the european electrical grid", *Energies*, vol. 13, no. 9, p. 2157, 2020.
- [5] G. Giruzzi *et al.*, "Model validation and integrated modelling simulations for the jt-60sa tokamak", 2012.
- [6] Culham Centre for Fusion Energy, *Fusion in brief*, <https://ccfe.ukaea.uk/fusion-energy/fusion-in-brief/>, Accessed: 2024-06-10, 2024.
- [7] A. Lukenskas *et al.*, "High heat flux test results for a thermal break demo divertor target and subsequent design and manufacture development", *Fusion Engineering and Design*, vol. 146, pp. 1657–1660, 2019.
- [8] J. Pearl, J. Paterson, K. Flinders, N. Mantel and J.-H. You, "Cyclic medium heat flux testing of a wta lattice structure on the hive facility", *Fusion Engineering and Design*, vol. 194, p. 113 699, 2023.

- [9] M. Tindall *et al.*, "Towards a fusion component digital twin—virtual test and monitoring of components in chimera by systems simulation", *Fusion Engineering and Design*, vol. 191, p. 113 773, 2023.
- [10] T. Barrett *et al.*, "The chimera facility development programme", *Fusion Engineering and Design*, vol. 194, p. 113 689, 2023.
- [11] P. Daniell, "Lectures on cauchy's problem in linear partial differential equations. by j. hadamard. pp. viii+ 316. 15s. net. 1923.(per oxford university press.)", *The Mathematical Gazette*, vol. 12, no. 171, pp. 173–174, 1924.
- [12] H. Lee and I. S. Kang, "Neural algorithm for solving differential equations", en, *J. Comput. Phys.*, vol. 91, no. 1, pp. 110–131, Nov. 1990, ISSN: 0021-9991. DOI: [10.1016/0021-9991\(90\)90007-N](https://doi.org/10.1016/0021-9991(90)90007-N). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/002199919090007N> (visited on 23/05/2022).
- [13] I. Lagaris, A. Likas and D. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries", *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1041–1049, Sep. 2000, Conference Name: IEEE Transactions on Neural Networks, ISSN: 1941-0093. DOI: [10.1109/72.870037](https://doi.org/10.1109/72.870037).
- [14] I. Lagaris, A. Likas and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations", *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 987–1000, Sep. 1998, Conference Name: IEEE Transactions on Neural Networks, ISSN: 1941-0093. DOI: [10.1109/72.712178](https://doi.org/10.1109/72.712178).
- [15] A. Malek and R. Shekari Beidokhti, "Numerical solution for high order differential equations using a hybrid neural network—Optimization method", en, *Appl. Math. Comput.*, vol. 183, no. 1, pp. 260–271, Dec. 2006, ISSN: 0096-3003. DOI: [10.1016/j.amc.2006.05.068](https://doi.org/10.1016/j.amc.2006.05.068). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0096300306005583> (visited on 23/05/2022).
- [16] K. Rudd and S. Ferrari, "A constrained integration (CINT) approach to solving partial differential equations using artificial neural networks", en, *Neurocomputing*, vol. 155, pp. 277–285, May 2015, ISSN: 0925-2312. DOI: [10.1016/j.neucom.2014.11.058](https://doi.org/10.1016/j.neucom.2014.11.058). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S092523121401652X> (visited on 23/05/2022).

- [17] M. Raissi, P. Perdikaris and G. E. Karniadakis, "Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations", en, *SIAM J. Sci. Comput.*, Jan. 2018, Publisher: Society for Industrial and Applied Mathematics. DOI: [10.1137/17M1120762](https://doi.org/10.1137/17M1120762). [Online]. Available: <https://epubs.siam.org/doi/abs/10.1137/17M1120762> (visited on 17/03/2022).
- [18] M. Raissi, Z. Wang, M. S. Triantafyllou and G. E. Karniadakis, "Deep learning of vortex-induced vibrations", en, *J. Fluid Mech.*, vol. 861, pp. 119–137, Feb. 2019, Publisher: Cambridge University Press, ISSN: 0022-1120, 1469-7645. DOI: [10.1017/jfm.2018.872](https://doi.org/10.1017/jfm.2018.872). [Online]. Available: <https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/deep-learning-of-vortex-induced-vibrations/B7D9B152C42B7F1E895C1661F5A85881> (visited on 17/03/2022).
- [19] M. P. Bonkile, A. Awasthi, C. Lakshmi, V. Mukundan and V. S. Aswin, "A systematic literature review of Burgers' equation with recent advances", en, *Pramana - J Phys*, vol. 90, no. 6, p. 69, Apr. 2018, ISSN: 0973-7111. DOI: [10.1007/s12043-018-1559-4](https://doi.org/10.1007/s12043-018-1559-4). [Online]. Available: <https://doi.org/10.1007/s12043-018-1559-4> (visited on 24/05/2022).
- [20] M. Raissi, P. Perdikaris and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations", en, *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019, ISSN: 0021-9991. DOI: [10.1016/j.jcp.2018.10.045](https://doi.org/10.1016/j.jcp.2018.10.045). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999118307125> (visited on 31/01/2022).
- [21] M. De Florio, E. Schiassi, B. D. Ganapol and R. Furfaro, "Physics-Informed Neural Networks for rarefied-gas dynamics: Poiseuille flow in the BGK approximation", en, *Z. Angew. Math. Phys.*, vol. 73, no. 3, p. 126, May 2022, ISSN: 1420-9039. DOI: [10.1007/s00033-022-01767-z](https://doi.org/10.1007/s00033-022-01767-z). [Online]. Available: <https://doi.org/10.1007/s00033-022-01767-z> (visited on 10/06/2022).
- [22] M. De Florio, E. Schiassi and R. Furfaro, "Physics-informed neural networks and functional interpolation for stiff chemical kinetics", *Chaos*, vol. 32, no. 6, p. 063107, Jun. 2022, Publisher: American Institute of Physics, ISSN: 1054-1500. DOI: [10.1063/5.0086649](https://doi.org/10.1063/5.0086649). [Online]. Available: <https://aip.scitation.org/doi/full/10.1063/5.0086649> (visited on 10/06/2022).

- [23] M. Aliakbari, M. Mahmoudi, P. Vadasz and A. Arzani, "Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport solvers using physics-informed neural networks", en, *Int. J. Heat Fluid Flow*, vol. 96, p. 109 002, Aug. 2022, ISSN: 0142-727X. DOI: [10.1016/j.ijheatfluidflow.2022.109002](https://doi.org/10.1016/j.ijheatfluidflow.2022.109002). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0142727X22000777> (visited on 10/06/2022).
- [24] D. W. Abueidda, S. Koric, E. Guleryuz and N. A. Sobh, "Enhanced physics-informed neural networks for hyperelasticity", arXiv, Tech. Rep. arXiv:2205.14148, May 2022, arXiv:2205.14148 [cs] type: article. DOI: [10.48550/arXiv.2205.14148](https://doi.org/10.48550/arXiv.2205.14148). [Online]. Available: <http://arxiv.org/abs/2205.14148> (visited on 10/06/2022).
- [25] C. Xu, B. T. Cao, Y. Yuan and G. Meschke, "Transfer learning based physics-informed neural networks for solving inverse problems in tunneling", arXiv, Tech. Rep. arXiv:2205.07731, May 2022, arXiv:2205.07731 [cs] type: article. DOI: [10.48550/arXiv.2205.07731](https://doi.org/10.48550/arXiv.2205.07731). [Online]. Available: <http://arxiv.org/abs/2205.07731> (visited on 10/06/2022).
- [26] B. Zapf, J. Haubner, M. Kuchta, G. Ringstad, P. K. Eide and K.-A. Mardal, "Investigating molecular transport in the human brain from MRI with physics-informed neural networks", arXiv, Tech. Rep. arXiv:2205.02592, May 2022, arXiv:2205.02592 [cs, math] type: article. DOI: [10.48550/arXiv.2205.02592](https://doi.org/10.48550/arXiv.2205.02592). [Online]. Available: <http://arxiv.org/abs/2205.02592> (visited on 10/06/2022).
- [27] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo and S. G. Johnson, "Physics-Informed Neural Networks with Hard Constraints for Inverse Design", *SIAM J. Sci. Comput.*, vol. 43, no. 6, B1105–B1132, Jan. 2021, Publisher: Society for Industrial and Applied Mathematics, ISSN: 1064-8275. DOI: [10.1137/21M1397908](https://doi.org/10.1137/21M1397908). [Online]. Available: <https://epubs.siam.org/doi/10.1137/21M1397908> (visited on 27/05/2022).
- [28] E. Stevens, L. Antiga and T. Viehmann, *Deep learning with PyTorch*. Manning Publications, 2020.
- [29] B. Pang, E. Nijkamp and Y. N. Wu, "Deep learning with tensorflow: A review", *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, 2020.
- [30] C. C. Margossian, "A review of automatic differentiation and its efficient implementation", en, *WIREs Data Min. Knowl. Discovery*, vol. 9, no. 4, e1305, 2019, _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1305>, ISSN: 1942-4795. DOI: [10.](https://doi.org/10.1002/widm.1305)

- 1002/widm.1305. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1305> (visited on 23/05/2022).
- [31] H. Robinson, S. Pawar, A. Rasheed and O. San, "Physics guided neural networks for modelling of non-linear dynamics", *Neural Networks*, vol. 154, pp. 333–345, 2022.
- [32] K. Andersen, G. E. Cook, G. Karsai and K. Ramaswamy, "Artificial neural networks applied to arc welding process modeling and control", *IEEE Transactions on industry applications*, vol. 26, no. 5, pp. 824–830, 1990.
- [33] C. Rao, H. Sun and Y. Liu, "Hard encoding of physics for learning spatiotemporal dynamics", *arXiv preprint arXiv:2105.00557*, 2021.
- [34] R. T. Chen, Y. Rubanova, J. Bettencourt and D. K. Duvenaud, "Neural ordinary differential equations", *Advances in neural information processing systems*, vol. 31, 2018.
- [35] O. Zienkiewicz, R. Taylor and P. Nithiarasu, *The Finite Element Method for Fluid Dynamics*, 7th. Oxford: Elsevier, 2013.
- [36] P. Nithiarasu, R. Lewis and K. Seetharamu, *Fundamentals of the finite element method for heat and mass transfer*. Wiley, 2015.
- [37] C. Hirsch, *Numerical Computation of Internal and External Flows*. New York: John Wiley & Sons, 1992, vol. 1.
- [38] Y.-c. Wu and J.-w. Feng, "Development and Application of Artificial Neural Network", en, *Wireless Pers. Commun.*, vol. 102, no. 2, pp. 1645–1656, Sep. 2018, ISSN: 1572-834X. DOI: [10.1007/s11277-017-5224-x](https://doi.org/10.1007/s11277-017-5224-x). [Online]. Available: <https://doi.org/10.1007/s11277-017-5224-x> (visited on 01/08/2022).
- [39] D. Trehan, "Non-Convex Optimization: A Review", in *(ICICCS)*, May 2020, pp. 418–423. DOI: [10.1109/ICICCS48265.2020.9120874](https://doi.org/10.1109/ICICCS48265.2020.9120874).
- [40] T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems", *IEEE Trans. Neural Netw.*, vol. 6, no. 4, pp. 911–917, Jul. 1995, Conference Name: IEEE Transactions on Neural Networks, ISSN: 1941-0093. DOI: [10.1109/72.392253](https://doi.org/10.1109/72.392253).
- [41] D. Freedman, *Statistical Models: Theory and Practice*, en. Cambridge Press, Apr. 2009, Google-Books-ID: 4N3KOEitRe8C, ISBN: 978-0-521-11243-7.

- [42] W. C. Thacker, "The role of the Hessian matrix in fitting models to measurements", en, *J. Geophys. Res.: Oceans*, vol. 94, no. C5, pp. 6177–6196, 1989, ISSN: 2156-2202. DOI: [10.1029/JC094iC05p06177](https://doi.org/10.1029/JC094iC05p06177). [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1029/JC094iC05p06177> (visited on 28/03/2022).
- [43] P. Diaconis and M. Shahshahani, "On Nonlinear Functions of Linear Combinations", *SIAM J. Sci. Stat. Comput.*, vol. 5, no. 1, pp. 175–191, Mar. 1984, Publisher: Society for Industrial and Applied Mathematics, ISSN: 0196-5204. DOI: [10.1137/0905013](https://doi.org/10.1137/0905013). [Online]. Available: <https://epubs.siam.org/doi/abs/10.1137/0905013> (visited on 28/03/2022).
- [44] P. WERBOS, "Beyond Regression : New Tools for Prediction and Analysis in the Behavior Science", *Doctoral Dissertation, Harvard University*, 1974. [Online]. Available: <https://ci.nii.ac.jp/naid/10012540025/> (visited on 28/03/2022).
- [45] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*, en. MIT Press, 2017, ISBN: 978-0-262-34393-0. DOI: [10.7551/mitpress/11301.001.0001](https://doi.org/10.7551/mitpress/11301.001.0001). [Online]. Available: <https://direct.mit.edu/books/book/3132/perceptronsan-introduction-to-computational> (visited on 28/03/2022).
- [46] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020, ISBN: 0-244-76852-8.
- [47] Q. Wang, Y. Ma, K. Zhao and Y. Tian, "A Comprehensive Survey of Loss Functions in Machine Learning", en, *Ann. Data Sci.*, vol. 9, no. 2, pp. 187–212, Apr. 2022, ISSN: 2198-5812. DOI: [10.1007/s40745-020-00253-5](https://doi.org/10.1007/s40745-020-00253-5). [Online]. Available: <https://doi.org/10.1007/s40745-020-00253-5> (visited on 01/08/2022).
- [48] R. Elshaw, A. Wahab, A. Barnawi and S. Sakr, "DLBench: A comprehensive experimental evaluation of deep learning frameworks", en, *Cluster Comput.*, vol. 24, no. 3, pp. 2017–2038, Sep. 2021, ISSN: 1573-7543. DOI: [10.1007/s10586-021-03240-4](https://doi.org/10.1007/s10586-021-03240-4). [Online]. Available: <https://doi.org/10.1007/s10586-021-03240-4> (visited on 01/08/2022).
- [49] B. Ghogh, A. Ghodsi, F. Karray and M. Crowley, *KKT Conditions, First-Order and Second-Order Optimization, and Distributed Optimization: Tutorial and Survey*, arXiv:2110.01858 [cs, math], Oct. 2021. DOI: [10.48550/arXiv.2110.01858](https://doi.org/10.48550/arXiv.2110.01858). [Online]. Available: <http://arxiv.org/abs/2110.01858> (visited on 01/08/2022).

- [50] N. Ketkar, "Stochastic Gradient Descent", en, in *Deep Learning with Python: A Hands-on Introduction*, N. Ketkar, Ed., Berkeley, CA: apress, 2017, pp. 113–132, ISBN: 978-1-4842-2766-4. DOI: [10.1007/978-1-4842-2766-4_8](https://doi.org/10.1007/978-1-4842-2766-4_8). [Online]. Available: https://doi.org/10.1007/978-1-4842-2766-4_8 (visited on 01/08/2022).
- [51] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", arXiv, Tech. Rep. arXiv:1412.6980, Jan. 2017, arXiv:1412.6980 [cs] type: article. DOI: [10.48550/arXiv.1412.6980](https://arxiv.org/abs/1412.6980). [Online]. Available: <http://arxiv.org/abs/1412.6980> (visited on 26/05/2022).
- [52] R. Fletcher, "An Overview of Unconstrained Optimization", en, in *Algorithms for Continuous Optimization: The State of the Art*, ser. NATO ASI Series, E. Spedicato, Ed., Dordrecht: Springer Netherlands, 1994, pp. 109–143, ISBN: 978-94-009-0369-2. DOI: [10.1007/978-94-009-0369-2_5](https://doi.org/10.1007/978-94-009-0369-2_5). [Online]. Available: https://doi.org/10.1007/978-94-009-0369-2_5 (visited on 24/05/2022).
- [53] A. Ramm and A. Smirnova, "On stable numerical differentiation", en, *Math. Comput.*, vol. 70, no. 235, pp. 1131–1153, 2001, ISSN: 0025-5718, 1088-6842. DOI: [10.1090/S0025-5718-01-01307-2](https://www.ams.org/mcom/2001-70-235/S0025-5718-01-01307-2). [Online]. Available: [https://www.ams.org/mcom/2001-70-235/S0025-5718-01-01307-2/](https://www.ams.org/mcom/2001-70-235/S0025-5718-01-01307-2) (visited on 01/08/2022).
- [54] J. H. Davenport, Y. Siret and É. Tournier, *Computer algebra systems and algorithms for algebraic computation*. Acad. Press Prof., Inc., 1993, ISBN: 0-12-204232-8.
- [55] C. D. Barros, M. R. Mendonça, A. B. Vieira and A. Ziviani, "A survey on embedding dynamic graphs", *ACM Comput. Surv. (CSUR)*, vol. 55, no. 1, pp. 1–37, 2021, ISBN: 0360-0300 Publisher: ACM New York, NY.
- [56] B. Fang, E. Yang and F. Xie, "Symbolic Techniques for Deep Learning: Challenges and Opportunities", *arXiv preprint arXiv:2010.02727*, 2020.
- [57] M. Giles, "An extended collection of matrix derivative results for forward and reverse mode automatic differentiation", 2008.
- [58] R. Mathias, "A chain rule for matrix functions and applications", *SIAM J. Matrix Anal. Appl.*, vol. 17, no. 3, pp. 610–620, 1996, ISBN: 0895-4798 Publisher: SIAM.
- [59] S. Raschka, J. Patterson and C. Nolet, "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence", en, *Inf.*, vol. 11, no. 4, p. 193, Apr. 2020, Number: 4 Publisher: Multidisciplinary Digital Pub-

- lishing Institute, ISSN: 2078-2489. DOI: [10.3390/info11040193](https://doi.org/10.3390/info11040193). [Online]. Available: <https://www.mdpi.com/2078-2489/11/4/193> (visited on 25/05/2022).
- [60] R. Andonie, "Hyperparameter optimization in learning systems", en, *J. Membrane Comput.*, vol. 1, no. 4, pp. 279–291, Dec. 2019, ISSN: 2523-8914. DOI: [10.1007/s41965-019-00023-0](https://doi.org/10.1007/s41965-019-00023-0). [Online]. Available: <https://doi.org/10.1007/s41965-019-00023-0> (visited on 25/05/2022).
- [61] Y. Chen *et al.*, "Theory-guided hard constraint projection (HCP): A knowledge-based data-driven scientific machine learning method", en, *J. Comput. Phys.*, vol. 445, p. 110624, Nov. 2021, ISSN: 0021-9991. DOI: [10.1016/j.jcp.2021.110624](https://doi.org/10.1016/j.jcp.2021.110624). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999121005192> (visited on 17/03/2022).
- [62] G. Gnecco, M. Gori, S. Melacci and M. Sanguineti, "Learning with Hard Constraints", en, in *Artificial Neural Networks and Machine Learning – ICANN 2013*, V. Mladenov, P. Koprinkova-Hristova, G. Palm, A. E. P. Villa, B. Appollini and N. Kasabov, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: springer, 2013, pp. 146–153, ISBN: 978-3-642-40728-4. DOI: [10.1007/978-3-642-40728-4_19](https://doi.org/10.1007/978-3-642-40728-4_19).
- [63] A. Tato and R. Nkambou, "IMPROVING ADAM OPTIMIZER", en, p. 4, 2018.
- [64] V. Gopakumar, S. Pamela and D. Samaddar, "Loss Landscape Engineering via Data Regulation on PINNs", arXiv, Tech. Rep. arXiv:2205.07843, May 2022, arXiv:2205.07843 [physics] type: article. [Online]. Available: <http://arxiv.org/abs/2205.07843> (visited on 26/05/2022).
- [65] A. Arzani, J.-X. Wang and R. M. D'Souza, "Uncovering near-wall blood flow from sparse data with physics-informed neural networks", *Phys. Fluids*, vol. 33, no. 7, p. 071905, Jul. 2021, Publisher: American Institute of Physics, ISSN: 1070-6631. DOI: [10.1063/5.0055600](https://doi.org/10.1063/5.0055600). [Online]. Available: <https://aip.scitation.org/doi/abs/10.1063/5.0055600> (visited on 26/05/2022).
- [66] C. Bajaj, L. McLennan, T. Andeen and A. Roy, "Robust Learning of Physics Informed Neural Networks", arXiv, Tech. Rep. arXiv:2110.13330, Oct. 2021, arXiv:2110.13330 [cs, stat] type: article. DOI: [10.48550/arXiv.2110.13330](https://doi.org/10.48550/arXiv.2110.13330). [Online]. Available: <http://arxiv.org/abs/2110.13330> (visited on 26/05/2022).

- [67] M. A. Nabian, R. J. Gladstone and H. Meidani, "Efficient training of physics-informed neural networks via importance sampling", en, *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 36, no. 8, pp. 962–977, 2021, _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12685>, ISSN: 1467-8667. DOI: [10.1111/mice.12685](https://doi.org/10.1111/mice.12685). [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12685> (visited on 26/05/2022).
- [68] O. Hennigh *et al.*, "NVIDIA SimNet™: An AI-Accelerated Multi-Physics Simulation Framework", en, in *ICCS*, M. Paszynski, D. Kranzlmüller, V. V. Krzhizhanovskaya, J. J. Dongarra and P. M. Slood, Eds., ser. Lecture Notes in Computer Science, Cham: Springer Int. Pub., 2021, pp. 447–461, ISBN: 978-3-030-77977-1. DOI: [10.1007/978-3-030-77977-1_36](https://doi.org/10.1007/978-3-030-77977-1_36).
- [69] C. P. Robert and G. Casella, "Monte Carlo Integration", en, in *Monte Carlo Statistical Methods*, ser. Springer Texts in Statistics, C. P. Robert and G. Casella, Eds., New York, NY: springer, 1999, pp. 71–138, ISBN: 978-1-4757-3071-5. DOI: [10.1007/978-1-4757-3071-5_3](https://doi.org/10.1007/978-1-4757-3071-5_3). [Online]. Available: https://doi.org/10.1007/978-1-4757-3071-5_3 (visited on 26/05/2022).
- [70] W. J. Morokoff and R. E. Caflisch, "Quasi-Monte Carlo Integration", en, *J. Comput. Phys.*, vol. 122, no. 2, pp. 218–230, Dec. 1995, ISSN: 0021-9991. DOI: [10.1006/jcph.1995.1209](https://doi.org/10.1006/jcph.1995.1209). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999185712090> (visited on 26/05/2022).
- [71] S. Wang, Y. Teng and P. Perdikaris, "Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks", *SIAM J. Sci. Comput.*, vol. 43, no. 5, A3055–A3081, Jan. 2021, Publisher: Society for Industrial and Applied Mathematics, ISSN: 1064-8275. DOI: [10.1137/20M1318043](https://doi.org/10.1137/20M1318043). [Online]. Available: <https://epubs.siam.org/doi/abs/10.1137/20M1318043> (visited on 31/01/2022).
- [72] A. Vaswani *et al.*, "Attention is all you need", *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
- [73] S. Cao, "Choose a Transformer: Fourier or Galerkin", in *Adv. Neural Inf. Process. Syst.*, vol. 34, Curran Assoc., Inc., 2021, pp. 24 924–24 940.
- [74] H. Gao, M. J. Zahr and J.-X. Wang, "Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems", en, *Comput. Methods Appl. Mech. Eng.*, vol. 390, p. 114 502, Feb. 2022, ISSN: 0045-7825. DOI: [10.1016/j.cma.2021.114502](https://doi.org/10.1016/j.cma.2021.114502). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782521001145>

- [sciencedirect.com/science/article/pii/S0045782521007076](https://www.sciencedirect.com/science/article/pii/S0045782521007076) (visited on 01/08/2022).
- [75] J. Sirignano and K. Spiliopoulos, “DGM: A deep learning algorithm for solving partial differential equations”, en, *J. Comput. Phys.*, vol. 375, pp. 1339–1364, Dec. 2018, ISSN: 0021-9991. DOI: [10.1016/j.jcp.2018.08.029](https://doi.org/10.1016/j.jcp.2018.08.029). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999118305527> (visited on 23/05/2022).
- [76] Y. Yu, X. Si, C. Hu and J. Zhang, “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures”, *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, ISSN: 0899-7667. DOI: [10.1162/neco_a_01199](https://doi.org/10.1162/neco_a_01199). [Online]. Available: https://doi.org/10.1162/neco_a_01199 (visited on 23/05/2022).
- [77] N. Rahaman *et al.*, “On the Spectral Bias of Neural Networks”, en, in *Proc. 36th Int. Conf. Mach. Learn.*, ISSN: 2640-3498, PMLR, May 2019, pp. 5301–5310. [Online]. Available: <https://proceedings.mlr.press/v97/rahaman19a.html> (visited on 23/05/2022).
- [78] M. Tancik *et al.*, “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”, in *Adv. Neural Inf. Process. Syst.*, vol. 33, Curran Assoc., Inc., 2020, pp. 7537–7547. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/hash/55053683268957697aa39fba6f231c68-Abstract.html> (visited on 23/05/2022).
- [79] *Modulus User Guide*, release v21.06, Nov. 2021. [Online]. Available: <https://developer.nvidia.com/modulus-user-guide-v2106>.
- [80] V. Sitzmann, J. Martel, A. Bergman, D. Lindell and G. Wetzstein, “Implicit Neural Representations with Periodic Activation Functions”, in *Adv. Neural Inf. Process. Syst.*, vol. 33, Curran Assoc., Inc., 2020, pp. 7462–7473. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/hash/53c04118df112c13a8c34b38343b9c10-Abstract.html> (visited on 23/05/2022).
- [81] C. L. Zhao, “Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks”, en, *Comm. Comput. Phys.*, vol. 29, no. 3, Jul. 2020. DOI: [10.4208/cicp.OA-2020-0086](https://doi.org/10.4208/cicp.OA-2020-0086). [Online]. Available: <https://par.nsf.gov/biblio/10225320-solving-allen-cahn-cahn-hilliard-equations-using-adaptive-physics-informed-neural-networks> (visited on 17/03/2022).

- [82] L. McClenny and U. Braga-Neto, "Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism", en, AAAI-MLPS, Tech. Rep. 68, Feb. 2019, p. 1 478 744. DOI: [10.2172/1478744](https://doi.org/10.2172/1478744). [Online]. Available: http://ceur-ws.org/Vol-2964/article_68.pdf (visited on 31/01/2022).
- [83] S. Shi, D. Liu and Z. Zhao, "Non-Fourier Heat Conduction based on Self-Adaptive Weight Physics-Informed Neural Networks", in *2021 40th Chin. Control Conf. (CCC)*, ISSN: 1934-1768, Jul. 2021, pp. 8451–8456. DOI: [10.23919/CCC52363.2021.9550487](https://doi.org/10.23919/CCC52363.2021.9550487).
- [84] S. Wang, X. Yu and P. Perdikaris, "When and why PINNs fail to train: A neural tangent kernel perspective", en, *J. Comput. Phys.*, vol. 449, p. 110 768, Jan. 2022, ISSN: 0021-9991. DOI: [10.1016/j.jcp.2021.110768](https://doi.org/10.1016/j.jcp.2021.110768). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S002199912100663X> (visited on 17/03/2022).
- [85] R.-Y. Sun, "Optimization for Deep Learning: An Overview", en, *J. Oper. Res. Soc. China*, vol. 8, no. 2, pp. 249–294, Jun. 2020, ISSN: 2194-6698. DOI: [10.1007/s40305-020-00309-6](https://doi.org/10.1007/s40305-020-00309-6). [Online]. Available: <https://doi.org/10.1007/s40305-020-00309-6> (visited on 24/05/2022).
- [86] R. Pascanu, T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks", en, in *Proc. 30th Int. Conf. Mach. Learn.*, ISSN: 1938-7228, PMLR, May 2013, pp. 1310–1318. [Online]. Available: <https://proceedings.mlr.press/v28/pascanu13.html> (visited on 23/05/2022).
- [87] H. H. Tan and K. H. Lim, "Review of second-order optimization techniques in artificial neural networks backpropagation", en, *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 495, p. 012 003, Jun. 2019, Publisher: IOP Publishing, ISSN: 1757-899X. DOI: [10.1088/1757-899X/495/1/012003](https://doi.org/10.1088/1757-899X/495/1/012003). [Online]. Available: <https://doi.org/10.1088/1757-899x/495/1/012003> (visited on 24/05/2022).
- [88] C. Bard and J. Dorelli, "Neural Network Reconstruction of Plasma Space-Time", *Front. Astron. Space Sci.*, vol. 8, 2021, ISSN: 2296-987X. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fspas.2021.732275> (visited on 07/04/2022).
- [89] L. Martino, V. Elvira and F. Louzada, "Effective sample size for importance sampling based on discrepancy measures", en, *Signal Process.*, vol. 131, pp. 386–401, Feb. 2017, ISSN: 0165-1684. DOI: [10.1016/j.sigpro.2016.08.025](https://doi.org/10.1016/j.sigpro.2016.08.025). [Online]. Available: <https://>

- www.sciencedirect.com/science/article/pii/S0165168416302110 (visited on 26/05/2022).
- [90] P. Sharma, L. Evans, M. Tindall and P. Nithiarasu, "Hyperparameter selection for physics-informed neural networks (pinns)–application to discontinuous heat conduction problems", *Numerical Heat Transfer, Part B: Fundamentals*, pp. 1–15, 2023.
- [91] A. D. Jagtap, K. Kawaguchi and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks", en, *J. Comput. Phys.*, vol. 404, p. 109136, Mar. 2020, ISSN: 0021-9991. DOI: [10.1016/j.jcp.2019.109136](https://doi.org/10.1016/j.jcp.2019.109136). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999119308411> (visited on 18/03/2022).
- [92] A. D. Jagtap, K. Kawaguchi and G. Em Karniadakis, "Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks", *Proceedings of the Royal Society A*, vol. 476, no. 2239, p. 20200334, 2020.
- [93] E. Schiassi, C. Leake, M. De Florio, H. Johnston, R. Furfaro and D. Mortari, "Extreme Theory of Functional Connections: A Physics-Informed Neural Network Method for Solving Parametric Differential Equations", arXiv, Tech. Rep. arXiv:2005.10632, May 2020, arXiv:2005.10632 [physics, stat] type: article. DOI: [10.48550/arXiv.2005.10632](https://doi.org/10.48550/arXiv.2005.10632). [Online]. Available: <http://arxiv.org/abs/2005.10632> (visited on 26/05/2022).
- [94] S. Wang, H. Wang and P. Perdikaris, "Learning the solution operator of parametric partial differential equations with physics-informed DeepONets", *Sci. Adv.*, vol. 7, no. 40, eabi8605, Sep. 2021, Publisher: American Association for the Advancement of Science. DOI: [10.1126/sciadv.abi8605](https://doi.org/10.1126/sciadv.abi8605). [Online]. Available: <https://www.science.org/doi/10.1126/sciadv.abi8605> (visited on 27/05/2022).
- [95] S. Koric and D. W. Abueidda, "Data-driven and physics-informed deep learning operators for solution of heat conduction equation with parametric heat source", *International Journal of Heat and Mass Transfer*, vol. 203, p. 123809, 2023.
- [96] A. D. Jagtap, E. Kharazmi and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems", en, *Comput. Methods Appl. Mech. Eng.*, vol. 365, p. 113028, Jun. 2020, ISSN: 0045-7825. DOI: [10.1016/j.cma.2020.113028](https://doi.org/10.1016/j.cma.2020.113028). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782520302127> (visited on 17/03/2022).

- [97] A. D. Jagtap and G. Em Karniadakis, "Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for non-linear partial differential equations", *Communications in Computational Physics*, vol. 28, no. 5, pp. 2002–2041, 2020, ISSN: 1991-7120. DOI: <https://doi.org/10.4208/cicp.OA-2020-0164>. [Online]. Available: http://global-sci.org/intro/article_detail/cicp/18403.html.
- [98] K. Shukla, A. D. Jagtap and G. E. Karniadakis, "Parallel physics-informed neural networks via domain decomposition", *Journal of Computational Physics*, vol. 447, p. 110683, 2021.
- [99] Z. Hu, A. D. Jagtap, G. E. Karniadakis and K. Kawaguchi, "When Do Extended Physics-Informed Neural Networks (XPINNs) Improve Generalization?", *arXiv:2109.09444 [cs, math, stat]*, Dec. 2021, arXiv: 2109.09444. [Online]. Available: <http://arxiv.org/abs/2109.09444> (visited on 17/03/2022).
- [100] J. Xu and L. T. Zikatanov, "On multigrid methods for generalized finite element methods", in *Meshfree methods for partial differential equations*, Springer, 2003, pp. 401–418.
- [101] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.
- [102] E. Haghighat, M. Raissi, A. Moure, H. Gomez and R. Juanes, "A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics", in *Comput. Methods Appl. Mech. Eng.*, vol. 379, p. 113741, Jun. 2021, ISSN: 0045-7825. DOI: [10.1016/j.cma.2021.113741](https://doi.org/10.1016/j.cma.2021.113741). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782521000773> (visited on 17/03/2022).
- [103] K. Prantikos, S. Chatzidakis, L. H. Tsoukalas and A. Heifetz, "Physics-informed neural network with transfer learning (tl-pinn) based on domain similarity measure for prediction of nuclear reactor transients", *Scientific Reports*, vol. 13, no. 1, p. 16840, 2023.
- [104] A. Vaswani et al., "Attention is All you Need", in *Adv. Neural Inf. Process. Syst.*, vol. 30, Curran Assoc., Inc., 2017. [Online]. Available: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html> (visited on 27/05/2022).
- [105] T. F. Miller, *Application of multilevel methods to a semi-implicit pressure-linked algorithm*. The Pennsylvania State University, 1987.

- [106] *Stiff Differential Equations*, en. [Online]. Available: <https://uk.mathworks.com/company/newsletters/articles/stiff-differential-equations.html> (visited on 25/05/2022).
- [107] L. Lu, X. Meng, Z. Mao and G. E. Karniadakis, "DeepXDE: A Deep Learning Library for Solving Differential Equations", *SIAM Rev.*, vol. 63, no. 1, pp. 208–228, Jan. 2021, ISSN: 0036-1445. DOI: [10.1137/19M1274067](https://doi.org/10.1137/19M1274067). [Online]. Available: <https://epubs.siam.org/doi/10.1137/19M1274067> (visited on 17/03/2022).
- [108] E. Haghighat and R. Juanes, "SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks", en, *Comput. Methods Appl. Mech. Eng.*, vol. 373, p. 113552, Jan. 2021, ISSN: 0045-7825. DOI: [10.1016/j.cma.2020.113552](https://doi.org/10.1016/j.cma.2020.113552). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782520307374> (visited on 24/05/2022).
- [109] L. D. McClenny, M. A. Haile and U. M. Braga-Neto, "TensorDiffEq: Scalable Multi-GPU Forward and Inverse Solvers for Physics Informed Neural Networks", arXiv, Tech. Rep. arXiv:2103.16034, Mar. 2021, arXiv:2103.16034 [physics] type: article. [Online]. Available: <http://arxiv.org/abs/2103.16034> (visited on 24/05/2022).
- [110] K. Zubov *et al.*, "NeuralPDE: Automating Physics-Informed Neural Networks (PINNs) with Error Approximations", arXiv, Tech. Rep. arXiv:2107.09443, Jul. 2021, arXiv:2107.09443 [cs] type: article. [Online]. Available: <http://arxiv.org/abs/2107.09443> (visited on 24/05/2022).
- [111] N. Demo, M. Strazzullo and G. Rozza, "An extended physics informed neural network for preliminary analysis of parametric optimal control problems", arXiv, Tech. Rep. arXiv:2110.13530, Oct. 2021, arXiv:2110.13530 [cs, math] type: article. DOI: [10.48550/arXiv.2110.13530](https://doi.org/10.48550/arXiv.2110.13530). [Online]. Available: <http://arxiv.org/abs/2110.13530> (visited on 26/05/2022).
- [112] M. Raj, P. Kumbhar and R. K. Annabattula, "Physics-informed neural networks for solving thermo-mechanics problems of functionally graded material", arXiv, Tech. Rep. arXiv:2111.10751, Jan. 2022, arXiv:2111.10751 [cs] type: article. DOI: [10.48550/arXiv.2111.10751](https://doi.org/10.48550/arXiv.2111.10751). [Online]. Available: <http://arxiv.org/abs/2111.10751> (visited on 26/05/2022).

- [113] P. Heger, M. Full, D. Hilger and N. Hosters, "Investigation of Physics-Informed Deep Learning for the Prediction of Parametric, Three-Dimensional Flow Based on Boundary Data", arXiv, Tech. Rep. arXiv:2203.09204, Mar. 2022, arXiv:2203.09204 [cs, math] type: article. DOI: [10.48550/arXiv.2203.09204](https://doi.org/10.48550/arXiv.2203.09204). [Online]. Available: <http://arxiv.org/abs/2203.09204> (visited on 26/05/2022).
- [114] I. Berrada, J. A. Ferland and P. Michelon, "A multi-objective approach to nurse scheduling with both hard and soft constraints", en, *Socioecon. Plann. Sci.*, vol. 30, no. 3, pp. 183–193, Sep. 1996, ISSN: 0038-0121. DOI: [10.1016/0038-0121\(96\)00010-9](https://doi.org/10.1016/0038-0121(96)00010-9). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0038012196000109> (visited on 24/06/2022).
- [115] N. Sukumar and A. Srivastava, "Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks", en, *Comput. Methods Appl. Mech. Eng.*, vol. 389, p. 114 333, Feb. 2022, ISSN: 0045-7825. DOI: [10.1016/j.cma.2021.114333](https://doi.org/10.1016/j.cma.2021.114333). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782521006186> (visited on 26/05/2022).
- [116] H. Son, J. W. Jang, W. J. Han and H. J. Hwang, "Sobolev Training for Physics Informed Neural Networks", arXiv, Tech. Rep. arXiv:2101.08932, Dec. 2021, arXiv:2101.08932 [cs, math] type: article. DOI: [10.48550/arXiv.2101.08932](https://doi.org/10.48550/arXiv.2101.08932). [Online]. Available: <http://arxiv.org/abs/2101.08932> (visited on 26/05/2022).
- [117] J. Yu, L. Lu, X. Meng and G. E. Karniadakis, "Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems", *arXiv:2111.02801 [physics]*, Nov. 2021, arXiv: 2111.02801. [Online]. Available: <http://arxiv.org/abs/2111.02801> (visited on 17/03/2022).
- [118] T. Chan and W. Zhu, "Level set based shape prior segmentation", in *CVPR*, ISSN: 1063-6919, vol. 2, Jun. 2005, 1164–1170 vol. 2. DOI: [10.1109/CVPR.2005.212](https://doi.org/10.1109/CVPR.2005.212).
- [119] Z. Xiang, W. Peng, W. Zhou and W. Yao, "Hybrid Finite Difference with the Physics-informed Neural Network for solving PDE in complex geometries", *arXiv:2202.07926 [physics]*, Feb. 2022, arXiv: 2202.07926. [Online]. Available: <http://arxiv.org/abs/2202.07926> (visited on 17/03/2022).
- [120] M. R. Samsami and H. Alimadad, "Distributed Deep Reinforcement Learning: An Overview", arXiv, Tech. Rep. arXiv:2011.11012, Nov. 2020, arXiv:2011.11012 [cs] type: article. DOI: [10.48550/arXiv.2011.11012](https://doi.org/10.48550/arXiv.2011.11012). [Online]. Available: <http://arxiv.org/abs/2011.11012> (visited on 26/05/2022).

- [121] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey", *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017, Conference Name: IEEE Signal Processing Magazine, ISSN: 1558-0792. DOI: [10.1109/MSP.2017.2743240](https://doi.org/10.1109/MSP.2017.2743240).
- [122] A. Paszke *et al.*, "Automatic differentiation in pytorch", 2017.
- [123] M. Abadi, A. Agarwal and P. Barham, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: <https://www.tensorflow.org/>.
- [124] J. Bradbury *et al.*, *JAX: Composable transformations of Python+NumPy programs*, version 0.3.13, 2018. [Online]. Available: <http://github.com/google/jax>.
- [125] Y. Ma, D. Yu, T. Wu and H. Wang, "Paddlepaddle: An open-source deep learning platform from industrial practice", *Frontiers of Data and Computing*, vol. 1, no. 1, pp. 105–115, 2019.
- [126] T. Chen *et al.*, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems", *arXiv preprint arXiv:1512.01274*, 2015.
- [127] G. Pang, L. Lu and G. E. Karniadakis, "fPINNs: Fractional Physics-Informed Neural Networks", *SIAM J. Sci. Comput.*, vol. 41, no. 4, A2603–A2626, Jan. 2019, Publisher: Society for Industrial and Applied Mathematics, ISSN: 1064-8275. DOI: [10.1137/18M1229845](https://doi.org/10.1137/18M1229845). [Online]. Available: <https://epubs.siam.org/doi/10.1137/18M1229845> (visited on 27/05/2022).
- [128] D. Zhang, L. Guo and G. E. Karniadakis, "Learning in Modal Space: Solving Time-Dependent Stochastic PDEs Using Physics-Informed Neural Networks", *SIAM J. Sci. Comput.*, vol. 42, no. 2, A639–A665, Jan. 2020, Publisher: Society for Industrial and Applied Mathematics, ISSN: 1064-8275. DOI: [10.1137/19M1260141](https://doi.org/10.1137/19M1260141). [Online]. Available: <https://epubs.siam.org/doi/10.1137/19M1260141> (visited on 27/05/2022).
- [129] X. Meng and G. E. Karniadakis, "A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems", en, *J. Comput. Phys.*, vol. 401, p. 109020, Jan. 2020, ISSN: 0021-9991. DOI: [10.1016/j.jcp.2019.109020](https://doi.org/10.1016/j.jcp.2019.109020). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999119307260> (visited on 27/05/2022).

- [130] L. Lu, M. Dao, P. Kumar, U. Ramamurty, G. E. Karniadakis and S. Suresh, "Extraction of mechanical properties of materials through deep learning from instrumented indentation", *Proc. Natl. Acad. Sci.*, vol. 117, no. 13, pp. 7052–7062, Mar. 2020, Publisher: Proceedings of the National Academy of Sciences. DOI: [10.1073/pnas.1922210117](https://doi.org/10.1073/pnas.1922210117). [Online]. Available: <https://www.pnas.org/doi/full/10.1073/pnas.1922210117> (visited on 27/05/2022).
- [131] S. Wang, H. Wang and P. Perdikaris, "On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks", en, *Comput. Methods Appl. Mech. Eng.*, vol. 384, p. 113938, Oct. 2021, ISSN: 0045-7825. DOI: [10.1016/j.cma.2021.113938](https://doi.org/10.1016/j.cma.2021.113938). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782521002759> (visited on 27/05/2022).
- [132] L. Lu, P. Jin, G. Pang, Z. Zhang and G. E. Karniadakis, "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators", en, *Nat. Mach. Intell.*, vol. 3, no. 3, pp. 218–229, Mar. 2021, Number: 3 Publisher: Nature Publishing Group, ISSN: 2522-5839. DOI: [10.1038/s42256-021-00302-5](https://doi.org/10.1038/s42256-021-00302-5). [Online]. Available: <https://www.nature.com/articles/s42256-021-00302-5> (visited on 27/05/2022).
- [133] P. Jin, S. Meng and L. Lu, "MIONet: Learning multiple-input operators via tensor product", arXiv, Tech. Rep. arXiv:2202.06137, Feb. 2022, arXiv:2202.06137 [physics] type: article. DOI: [10.48550/arXiv.2202.06137](https://doi.org/10.48550/arXiv.2202.06137). [Online]. Available: <http://arxiv.org/abs/2202.06137> (visited on 27/05/2022).
- [134] S. Cai, Z. Wang, L. Lu, T. A. Zaki and G. E. Karniadakis, "DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks", en, *J. Comput. Phys.*, vol. 436, p. 110296, Jul. 2021, ISSN: 0021-9991. DOI: [10.1016/j.jcp.2021.110296](https://doi.org/10.1016/j.jcp.2021.110296). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999121001911> (visited on 27/05/2022).
- [135] Z. Mao, L. Lu, O. Marxen, T. A. Zaki and G. E. Karniadakis, "DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators", en, *J. Comput. Phys.*, vol. 447, p. 110698, Dec. 2021, ISSN: 0021-9991. DOI: [10.1016/j.jcp.2021.110698](https://doi.org/10.1016/j.jcp.2021.110698). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999121005933> (visited on 27/05/2022).

- [136] L. Lu, R. Pestourie, S. G. Johnson and G. Romano, "Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport", arXiv, Tech. Rep. arXiv:2204.06684, Apr. 2022, arXiv:2204.06684 [physics] type: article. DOI: [10.48550/arXiv.2204.06684](https://doi.org/10.48550/arXiv.2204.06684). [Online]. Available: <http://arxiv.org/abs/2204.06684> (visited on 27/05/2022).
- [137] R. K. Srivastava, K. Greff and J. Schmidhuber, "Training Very Deep Networks", in *Adv. Neural Inf. Process. Syst.*, vol. 28, Curran Assoc., Inc., 2015.
- [138] J. Guibas, M. Mardani, Z. Li, A. Tao, A. Anandkumar and B. Catanzaro, "Adaptive Fourier Neural Operators: Efficient Token Mixers for Transformers", arXiv, Tech. Rep. arXiv:2111.13587, Mar. 2022, arXiv:2111.13587 [cs] type: article. [Online]. Available: <http://arxiv.org/abs/2111.13587> (visited on 27/05/2022).
- [139] Z. Li *et al.*, "Physics-informed neural operator for learning partial differential equations", *arxiv preprint arxiv:2111.03794*, 2021.
- [140] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, "Image-to-image translation with conditional adversarial networks", in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1125–1134.
- [141] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans", in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8798–8807.
- [142] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network", in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4681–4690.
- [143] *Partial Differential Equation Toolbox (R2022a)*, en. [Online]. Available: <https://uk.mathworks.com/products/pde.html> (visited on 27/05/2022).
- [144] A. Lewkowycz, *How to decay your learning rate*, arXiv:2103.12682 [cs], Mar. 2021. DOI: [10.48550/arXiv.2103.12682](https://doi.org/10.48550/arXiv.2103.12682). [Online]. Available: <http://arxiv.org/abs/2103.12682> (visited on 25/08/2022).
- [145] S. Elfving, E. Uchibe and K. Doya, "Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning", arXiv, Tech. Rep. arXiv:1702.03118, Nov. 2017, arXiv:1702.03118 [cs] type: article. DOI: [10.48550/arXiv.1702.03118](https://doi.org/10.48550/arXiv.1702.03118). [Online]. Available: <http://arxiv.org/abs/1702.03118> (visited on 03/06/2022).

- [146] V. Schäfer, M. Bracke, R. Pinnau and M. Siggel, "Generalization of pinns for various boundary and initial conditions", Ph.D. dissertation, Master Thesis, University of Kaiserslautern, 2022.
- [147] E. Kharazmi, Z. Zhang and G. E. M. Karniadakis, "Hp-VPINNs: Variational physics-informed neural networks with domain decomposition", en, *Comput. Methods Appl. Mech. Eng.*, vol. 374, p. 113 547, Feb. 2021, ISSN: 0045-7825. DOI: [10.1016/j.cma.2020.113547](https://doi.org/10.1016/j.cma.2020.113547). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782520307325> (visited on 17/03/2022).
- [148] Y. Shin, "On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs", *Comm. Comput. Phys.*, vol. 28, no. 5, pp. 2042–2074, Jun. 2020, ISSN: 1815-2406, 1991-7120. DOI: [10.4208/cicp.OA-2020-0193](https://doi.org/10.4208/cicp.OA-2020-0193). [Online]. Available: http://global-sci.org/intro/article_detail/cicp/18404.html (visited on 25/07/2023).
- [149] S. Cuomo, V. S. di Cola, F. Giampaolo, G. Rozza, M. Raissi and F. Piccialli, "Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next", *arXiv:2201.05624 [physics]*, Feb. 2022, arXiv: 2201.05624. [Online]. Available: <http://arxiv.org/abs/2201.05624> (visited on 17/03/2022).
- [150] P. Sharma, L. Evans, M. Tindall and P. Nithiarasu, "Stiff-PDEs and Physics-Informed Neural Networks", en, *Arch. Comput. Methods Eng.*, Feb. 2023, ISSN: 1886-1784. DOI: [10.1007/s11831-023-09890-4](https://doi.org/10.1007/s11831-023-09890-4). [Online]. Available: <https://doi.org/10.1007/s11831-023-09890-4> (visited on 08/02/2023).
- [151] X.-K. Wen, G.-Z. Wu, W. Liu and C.-Q. Dai, "Dynamics of diverse data-driven solitons for the three-component coupled nonlinear Schrödinger model by the MPS-PINN method", en, *Nonlinear Dyn.*, vol. 109, no. 4, pp. 3041–3050, Sep. 2022, ISSN: 0924-090X, 1573-269X. DOI: [10.1007/s11071-022-07583-4](https://doi.org/10.1007/s11071-022-07583-4). [Online]. Available: <https://link.springer.com/10.1007/s11071-022-07583-4> (visited on 25/07/2023).
- [152] C. Wei and R. Ooka, "Indoor airflow field reconstruction using physics-informed neural network", en, *Build. Environ.*, vol. 242, p. 110 563, Aug. 2023, ISSN: 03601323. DOI: [10.1016/j.buildenv.2023.110563](https://doi.org/10.1016/j.buildenv.2023.110563). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0360132323005905> (visited on 25/07/2023).
- [153] P. Ren, C. Rao, Y. Liu, J.-X. Wang and H. Sun, "PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs", en, *Comput. Methods Appl. Mech. Eng.*, vol. 389, p. 114 399, Feb. 2022, ISSN: 0045-7825. DOI: [10.1016/j.cma.2021.114399](https://doi.org/10.1016/j.cma.2021.114399).

114399. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782521006514> (visited on 24/03/2022).
- [154] S. Markidis, "The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers?", *Front. Big Data*, vol. 4, p. 669097, Nov. 2021, ISSN: 2624-909X. DOI: [10.3389/fdata.2021.669097](https://doi.org/10.3389/fdata.2021.669097). [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fdata.2021.669097/full> (visited on 25/07/2023).
- [155] P. Escapil-Inchauspé and G. A. Ruz, "Hyper-parameter tuning of physics-informed neural networks: Application to Helmholtz problems", *ArXiv preprint*, 2022, Publisher: arXiv Version Number: 2. DOI: [10.48550/ARXIV.2205.06704](https://doi.org/10.48550/ARXIV.2205.06704). [Online]. Available: <https://arxiv.org/abs/2205.06704> (visited on 25/07/2023).
- [156] Y. Wang, X. Han, C.-Y. Chang, D. Zha, U. Braga-Neto and X. Hu, "Auto-PINN: Understanding and Optimizing Physics-Informed Neural Architecture", *ArXiv preprint*, 2022, Publisher: arXiv Version Number: 1. DOI: [10.48550/ARXIV.2205.13748](https://doi.org/10.48550/ARXIV.2205.13748). [Online]. Available: <https://arxiv.org/abs/2205.13748> (visited on 25/07/2023).
- [157] P. Pantidis, H. Eldababy, C. M. Tagle and M. E. Mobasher, "Error convergence and engineering-guided hyperparameter search of PINNs: Towards optimized I-FENN performance", en, *Comput. Methods Appl. Mech. Eng.*, vol. 414, p. 116160, Sep. 2023, ISSN: 00457825. DOI: [10.1016/j.cma.2023.116160](https://doi.org/10.1016/j.cma.2023.116160). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0045782523002840> (visited on 25/07/2023).
- [158] G. Cho, D. Zhu, J. J. Campbell and M. Wang, "An LSTM-PINN Hybrid Method to Estimate Lithium-Ion Battery Pack Temperature", *IEEE Access*, vol. 10, pp. 100594–100604, 2022, ISSN: 2169-3536. DOI: [10.1109/ACCESS.2022.3208103](https://doi.org/10.1109/ACCESS.2022.3208103). [Online]. Available: <https://ieeexplore.ieee.org/document/9895422/> (visited on 25/07/2023).
- [159] N. Hasebrook, F. Morsbach, N. Kannengießer, J. Franke, F. Hutter and A. Sunyaev, *Why Do Machine Learning Practitioners Still Use Manual Tuning? A Qualitative Study*, arXiv:2203.01717 [cs], Mar. 2022. DOI: [10.48550/arXiv.2203.01717](https://doi.org/10.48550/arXiv.2203.01717). [Online]. Available: <http://arxiv.org/abs/2203.01717> (visited on 02/05/2023).
- [160] A. Apicella, F. Donnarumma, F. Isgrò and R. Prevete, "A survey on modern trainable activation functions", *Neural Networks*, vol. 138, pp. 14–32, 2021.

- [161] D. Bertoin, J. Bolte, S. Gerchinovitz and E. Pauwels, “Numerical influence of ReLU’(0) on backpropagation”, in *Adv. Neural Inf. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang and J. W. Vaughan, Eds., vol. 34, Curran Assoc., Inc., 2021, pp. 468–479. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/043ab21fc5a1607b381ac3896176dac6-Paper.pdf.
- [162] H. Nishikawa, “On hyperbolic method for diffusion with discontinuous coefficients”, en, *J. Comput. Phys.*, vol. 367, pp. 102–108, Aug. 2018, ISSN: 00219991. DOI: [10.1016/j.jcp.2018.04.027](https://doi.org/10.1016/j.jcp.2018.04.027). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0021999118302444> (visited on 25/07/2023).
- [163] G. Klambauer, T. Unterthiner, A. Mayr and S. Hochreiter, *Self-Normalizing Neural Networks*, arXiv:1706.02515 [cs, stat], Sep. 2017. DOI: [10.48550/arXiv.1706.02515](https://doi.org/10.48550/arXiv.1706.02515). [Online]. Available: <http://arxiv.org/abs/1706.02515> (visited on 03/04/2023).
- [164] M. N. Özisik and H. R. B. Orlande, *Inverse Heat Transfer: Fundamentals and Applications*, en, 1st ed. Routledge, May 2018, ISBN: 978-0-203-74978-4. DOI: [10.1201/9780203749784](https://doi.org/10.1201/9780203749784). [Online]. Available: <https://www.taylorfrancis.com/books/9781351436403> (visited on 31/01/2023).
- [165] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh and Y. Villanger, “Local search: Is brute-force avoidable?”, en, *J. Comput. Syst. Sci.*, In Commemoration of Amir Pnueli, vol. 78, no. 3, pp. 707–719, May 2012, ISSN: 0022-0000. DOI: [10.1016/j.jcss.2011.10.003](https://doi.org/10.1016/j.jcss.2011.10.003). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0022000011001097> (visited on 31/01/2023).
- [166] R. Dechter and J. Pearl, “Generalized best-first search strategies and the optimality of A*”, en, *J. ACM*, vol. 32, no. 3, pp. 505–536, Jul. 1985, ISSN: 0004-5411, 1557-735X. DOI: [10.1145/3828.3830](https://doi.org/10.1145/3828.3830). [Online]. Available: <https://dl.acm.org/doi/10.1145/3828.3830> (visited on 31/01/2023).
- [167] A. N. Tikhonov, A. Goncharsky, V. V. Stepanov and A. G. Yagola, *Numerical Methods for the Solution of Ill-Posed Problems*, en. Springer Sci. Bus. Media, Jun. 1995, Google-Books-ID: rpdAzBsOSMgC, ISBN: 978-0-7923-3583-2.
- [168] H. W. Engl, M. Hanke and A. Neubauer, *Regularization of Inverse Problems*, en. Springer Sci. Bus. Media, Jul. 1996, Google-Books-ID: 2bzgmMv5EVcC, ISBN: 978-0-7923-4157-4.

- [169] Y. Jaluria, "Solution of Inverse Problems in Thermal Systems", *J. Therm. Sci. Eng. Appl.*, vol. 12, no. 1, Sep. 2019, ISSN: 1948-5085. DOI: [10.1115/1.4042353](https://doi.org/10.1115/1.4042353). [Online]. Available: <https://doi.org/10.1115/1.4042353> (visited on 31/01/2023).
- [170] Y. Jaluria, "Search methods", *Design and Optimization of Thermal Systems, 2nd ed.*; CRC Press: Boca Raton, FL, USA, pp. 511–558, 2008.
- [171] A. Bangian-Tabrizi and Y. Jaluria, "An optimization strategy for the inverse solution of a convection heat transfer problem", en, *Int. J. Heat Mass Transfer*, vol. 124, pp. 1147–1155, Sep. 2018, ISSN: 0017-9310. DOI: [10.1016/j.ijheatmasstransfer.2018.04.053](https://doi.org/10.1016/j.ijheatmasstransfer.2018.04.053). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0017931017344617> (visited on 31/01/2023).
- [172] F. Yaman, V. G. Yakhno and R. Potthast, "A Survey on Inverse Problems for Applied Sciences", en, *Math. Probl. Eng.*, vol. 2013, e976837, Jul. 2013, Publisher: Hindawi, ISSN: 1024-123X. DOI: [10.1155/2013/976837](https://doi.org/10.1155/2013/976837). [Online]. Available: <https://www.hindawi.com/journals/mpe/2013/976837/> (visited on 31/01/2023).
- [173] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators", en, *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989, ISSN: 0893-6080. DOI: [10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0893608089900208> (visited on 31/01/2023).
- [174] K. Hornik, "Approximation capabilities of multilayer feedforward networks", en, *Neural Netw.*, vol. 4, no. 2, pp. 251–257, Jan. 1991, ISSN: 0893-6080. DOI: [10.1016/0893-6080\(91\)90009-T](https://doi.org/10.1016/0893-6080(91)90009-T). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/089360809190009T> (visited on 31/01/2023).
- [175] K.-T. Yang, "Artificial Neural Networks (ANNs): A New Paradigm for Thermal Science and Engineering", *J. Heat Transfer*, vol. 130, no. 9, Jul. 2008, ISSN: 0022-1481. DOI: [10.1115/1.2944238](https://doi.org/10.1115/1.2944238). [Online]. Available: <https://doi.org/10.1115/1.2944238> (visited on 31/01/2023).
- [176] X. Guo, W. Li and F. Iorio, "Convolutional Neural Networks for Steady Flow Approximation", in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, ser. KDD '16, New York, NY, USA: ACM, Aug. 2016, pp. 481–490, ISBN: 978-1-4503-4232-2. DOI: [10.1145/2939672.2939738](https://doi.org/10.1145/2939672.2939738). [Online]. Available: <https://doi.org/10.1145/2939672.2939738> (visited on 31/01/2023).

- [177] H. R. Tamaddon-Jahromi, N. K. Chakshu, I. Sazonov, L. M. Evans, H. Thomas and P. Nithiarasu, "Data-driven inverse modelling through neural network (deep learning) and computational heat transfer", en, *Comput. Methods Appl. Mech. Eng.*, vol. 369, p. 113 217, Sep. 2020, ISSN: 0045-7825. DOI: [10.1016/j.cma.2020.113217](https://doi.org/10.1016/j.cma.2020.113217). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782520304023> (visited on 28/06/2022).
- [178] C. Xu, B. T. Cao, Y. Yuan and G. Meschke, "Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios", en, *Comput. Methods Appl. Mech. Eng.*, vol. 405, p. 115 852, Feb. 2023, ISSN: 0045-7825. DOI: [10.1016/j.cma.2022.115852](https://doi.org/10.1016/j.cma.2022.115852). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782522008088> (visited on 07/03/2023).
- [179] S. Cai, Z. Wang, S. Wang, P. Perdikaris and G. E. Karniadakis, "Physics-Informed Neural Networks for Heat Transfer Problems", *J. Heat Transfer*, vol. 143, no. 6, Apr. 2021, ISSN: 0022-1481. DOI: [10.1115/1.4050542](https://doi.org/10.1115/1.4050542). [Online]. Available: <https://doi.org/10.1115/1.4050542> (visited on 17/03/2022).
- [180] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang and L. Yang, "Physics-informed machine learning", en, *Nat. Rev. Phys.*, vol. 3, no. 6, pp. 422–440, Jun. 2021, Number: 6 Publisher: Nature Publishing Group, ISSN: 2522-5820. DOI: [10.1038/s42254-021-00314-5](https://doi.org/10.1038/s42254-021-00314-5). [Online]. Available: <https://www.nature.com/articles/s42254-021-00314-5> (visited on 17/03/2022).
- [181] A. F. Psaros, K. Kawaguchi and G. E. Karniadakis, "Meta-learning PINN loss functions", en, *J. Comput. Phys.*, vol. 458, p. 111 121, Jun. 2022, ISSN: 0021-9991. DOI: [10.1016/j.jcp.2022.111121](https://doi.org/10.1016/j.jcp.2022.111121). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999122001838> (visited on 17/03/2022).
- [182] Y. Chen *et al.*, "Physics-informed neural networks for inverse problems in nano-optics and metamaterials", EN, *Opt. Express*, vol. 28, no. 8, pp. 11 618–11 633, Apr. 2020, Publisher: Optica Publishing Group, ISSN: 1094-4087. DOI: [10.1364/OE.384875](https://doi.org/10.1364/OE.384875). [Online]. Available: <https://opg.optica.org/oe/abstract.cfm?uri=oe-28-8-11618> (visited on 10/06/2022).
- [183] X. Chen, L. Yang, J. Duan and G. E. Karniadakis, "Solving Inverse Stochastic Problems from Discrete Particle Observations Using the Fokker–Planck Equation and Physics-Informed Neural Networks", *SIAM J. Sci. Comput.*, vol. 43, no. 3, B811–B830, Jan.

- 2021, Publisher: Society for Industrial and Applied Mathematics, ISSN: 1064-8275. DOI: [10.1137/20M1360153](https://doi.org/10.1137/20M1360153). [Online]. Available: <https://epubs.siam.org/doi/abs/10.1137/20M1360153> (visited on 10/06/2022).
- [184] M. Raissi, A. Yazdani and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations", *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.
- [185] M.-S. Go, J. H. Lim and S. Lee, "Physics-informed neural network-based surrogate model for a virtual thermal sensor with real-time simulation", *International Journal of Heat and Mass Transfer*, vol. 214, p. 124 392, 2023.
- [186] N. Wang, H. Chang and D. Zhang, "Deep-learning-based inverse modeling approaches: A subsurface flow example", *Journal of Geophysical Research: Solid Earth*, vol. 126, no. 2, e2020JB020549, 2021.
- [187] P. Moser, W. Fenz, S. Thumfart, I. Ganitzer and M. Giretzlehner, "Modeling of 3d blood flows with physics-informed neural networks: Comparison of network architectures", *Fluids*, vol. 8, no. 2, p. 46, 2023.
- [188] F. Pioch, J. H. Harmening, A. M. Müller, F.-J. Peitzmann, D. Schramm and O. e. Moctar, "Turbulence modeling for physics-informed neural networks: Comparison of different rans models for the backward-facing step flow", *Fluids*, vol. 8, no. 2, p. 43, 2023.
- [189] S. L. Brunton, J. L. Proctor and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems", *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.
- [190] F. A. C. Viana, "A Tutorial on Latin Hypercube Design of Experiments", en, *Qual. Reliab. Eng. Int.*, vol. 32, no. 5, pp. 1975–1985, 2016, ISSN: 1099-1638. DOI: [10.1002/qre.1924](https://doi.org/10.1002/qre.1924). [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.1924> (visited on 27/05/2022).
- [191] K. Flinders *et al.*, "Characterising the impact of castellations on the efficiency of induction heating during testing in the hive facility", *Fusion Engineering and Design*, vol. 146, pp. 2040–2044, 2019.
- [192] D. Hancock, "Employing additive manufacturing for fusion high heat flux structures", Ph.D. dissertation, University of Sheffield, 2018.

- [193] D. Hancock *et al.*, “Testing advanced divertor concepts for fusion power plants using a small high heat flux facility”, 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:201748657>.
- [194] R. Lewis, L. Evans and B. Thorpe, *VirtualLab Documentation*, <https://virtuallab.readthedocs.io/en/dev/index.html>, Accessed: 2024-07-13, 2020.
- [195] G. M. Kurtzer, V. Sochat and M. W. Bauer, “Singularity: Scientific containers for mobility of compute”, *PloS one*, vol. 12, no. 5, e0177459, 2017.
- [196] V. Pakhnenko, “Application of the salome software package for numerical modeling geophysical tasks”, *Processes in GeoMedia—Volume IV*, pp. 375–385, 2022.
- [197] M. Andersson, L. Herrnsdorf and S. Mattsson, “Idac-cad2vox: A software for radiation protection calculations for the icrp reference phantoms which are in an environment described by detailed cad drawings”, in *MEDICAL PHYSICS IN THE BALTIC STATES: 13th International Conference on Medical Physics*, Kaunas Univ Technology Press, 2017, pp. 10–13.
- [198] M. R. Fathi, H. Latifi, H. Gholizadeh and S. Khare, “Paravis: An automatic python graphical package for ensemble analysis of plant beta diversity using remote sensing proxies”, *Ecological Informatics*, vol. 82, p. 102739, 2024.
- [199] Evans, Llion, *IBSim Website*, <https://ibsim.co.uk/>, Accessed: 2024-07-13, 2024.
- [200] R. Lewis, “Simulation driven machine learning methods to optimise design of physical experiments and enhance data analysis for testing of fusion energy heat exchanger components”, Ph.D. dissertation, Swansea University, 2024.
- [201] R. Otin, “Regularized maxwell equations and nodal finite elements for electromagnetic field computations”, *Electromagnetics*, vol. 30, no. 1-2, pp. 190–204, 2010.
- [202] E. M. Abo-Zahhad *et al.*, “Flow boiling in a four-compartment heat sink for high-heat flux cooling: A parametric study”, *Energy Conversion and Management*, vol. 230, p. 113778, 2021.
- [203] J. Delbecq, “The aster code; code aster”, 1999.
- [204] T. Gonda, M. Yakabe, Y. Hayamizu, S. Morita, T. Kawabe and K. Nishinaka, “Mechanical design using open-source software (eigenvalue analysis by parametric study)”, *Advances in Science and Technology*, vol. 139, pp. 19–24, 2024.

- [205] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent and J. Koster, "Mumps: A general purpose distributed memory sparse solver", in *International Workshop on Applied Parallel Computing*, Springer, 2000, pp. 121–130.
- [206] D. K. Ravikumar, Y. Than, W. Xu and J. Longtin, "Thermal considerations in the cryogenic regime for the bnl double ridge higher order mode waveguide", *Physical Review Accelerators and Beams*, vol. 20, no. 9, p. 093201, 2017.
- [207] L. Yang, X. Meng and G. E. Karniadakis, "B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data", *Journal of Computational Physics*, vol. 425, p. 109913, 2021.
- [208] S. Wang, H. Wang and P. Perdikaris, "Learning the solution operator of parametric partial differential equations with physics-informed deeponets", *Science advances*, vol. 7, no. 40, eabi8605, 2021.