Diss. ETH No. 29849

Applications of Deep Learning to Scientific Computing

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
ROBERTO MOLINARO
M.Sc. Mechanical Engineering, ETH Zurich
born June 16, 1995
citizen of Italy

Examination Committee
Prof. Dr. Siddhartha Mishra, ETH Zürich, examiner
Prof. Dr. George Karniadakis, Brown University, co-examiner
Prof. Dr. Paris Perdikaris, University of Pennsylvania, co-examiner

2023

Abstract

Physics-informed neural networks (PINNs) have been widely used for the robust and accurate approximation of partial differential equations. In the present thesis, we provide upper bounds on the generalization error of PINNs approximating solutions to the forward and inverse problems for PDEs. Specifically, we focus on a particular class of inverse problems, the so-called data assimilation or unique continuation problems. An abstract formalism is introduced, and stability properties of the underlying PDE are leveraged to derive an estimate for the generalization error in terms of the training error and the number of training samples. This abstract framework is illustrated with several examples of PDEs, and numerical examples validating the proposed theory are also presented. The derived estimates show two relevant facts: (1) PINNs require regularity of solutions to the underlying PDE to guarantee accurate approximation. Consequently, they may fail to approximate discontinuous solutions of PDEs, such as nonlinear hyperbolic equations. We then propose a novel variant of PINNs, termed weak PINNs (wPINNs), for accurate approximation of entropy solutions of scalar conservation laws. wPINNs are based on approximating the solution of a min-max optimization problem for a residual, defined in terms of Kruzhkov entropies, to determine parameters for the neural networks approximating the entropy solution as well as test functions. Moreover, (2) with a suitable quadrature rule, i.e., Monte Carlo quadrature, PINNs may potentially overcome the curse of dimensionality. Hence, we embrace physics-informed neural networks (PINNs) to solve the forward and inverse problems for a broad range of high-dimensional PDEs, including the radiative transfer equation and financial equations. We present a suite of numerical experiments demonstrating that PINNs provide very accurate solutions for both the forward and inverse problems at low computational cost without incurring the curse of dimensionality. In the final part of the thesis, we transition to the operator learning framework and consider a class of inverse problems for PDEs that are only well-defined as mappings from operators to functions. Existing operator learning architectures map functions to functions and need to be modified to learn inverse maps from data. We propose a novel architecture termed Neural Inverse Operators (NIOs) to solve these PDE inverse problems. Motivated by the underlying mathematical structure, NIO is based on a suitable composition of DeepONets and FNOs to approximate mappings from operators to functions. A variety of experiments are presented to demonstrate that NIO significantly outperform baselines and solve PDE inverse problems robustly and accurately. Moreover, NIO is several orders of magnitude faster than existing direct and PDE-constrained optimization methods.

Sinossi

Le reti neurali informate dalla fisica (PINNs) sono state ampiamente utilizzate per approssimare con robustezza e precisione le equazioni differenziali alle derivate parziali. In questa tesi, forniamo limiti superiori sull'errore di generalizzazione delle PINNs nel calcolo delle soluzioni di problemi diretti e inversi per le EDP. Ci concentriamo su una classe particolare di problemi inversi, noti come problemi di assimilazione dei dati o di continuazione unica. Presentiamo un formalismo astratto e sfruttiamo le proprietà di stabilità delle EDP corrispondenti per derivare una stima dell'errore di generalizzazione in funzione dell'errore di addestramento e del numero di esempi di addestramento. Questo quadro astratto è illustrato con vari esempi di EDP e sono presentati anche esempi numerici che validano la teoria proposta. Le stime ottenute mostrano due fatti rilevanti: (1) le PINNs richiedono la regolarità delle soluzioni alle EDP sottostanti per garantire un'approssimazione accurata. Di conseguenza, potrebbero non riuscire ad approssimare soluzioni discontinue di EDP, come le equazioni iperboliche non lineari. Proponiamo quindi una nuova variante delle PINNs, denominate wPINNs, per l'approssimazione accurata delle soluzioni entropiche delle leggi di conservazione scalare. Le wPINNs si basano sull'approssimazione della soluzione di un problema di ottimizzazione min-max per un residuo definito in termini di entropie di Kruzhkov, per determinare i parametri delle reti neurali che approssimano la soluzione entropica così come le funzioni di test. Inoltre, (2) con un metodo di integrazione adeguato, come l'integrazione di Monte Carlo, le PINNs potrebbero potenzialmente superare la *curse of dimensionality.* Pertanto, adottiamo le reti neurali informate dalla fisica (PINNs) per risolvere i problemi diretti e inversi per un'ampia gamma di EDP ad alta dimensionalità, comprese l'equazione del trasferimento radiativo e le equazioni finanziarie. Presentiamo una serie di esperimenti numerici che dimostrano come le PINNs forniscano soluzioni molto accurate per entrambi i problemi diretti e inversi a basso costo computazionale senza incorrere nella *curse of dimensionality.*

Nella parte finale della tesi, passiamo al framework di *operator learning* e consideriamo una classe di problemi inversi per le EDP che sono ben definiti solo come mappe da operatori a funzioni. Le attuali architetture di apprendimento degli operatori mappano funzioni a funzioni e devono essere modificate per imparare mappe inverse dai dati. Proponiamo una nuova architettura chiamata Neural Inverse Operator (NIO) per risolvere questi problemi inversi delle EDP. Motivati dalla struttura matematica sottostante, NIO si basa su una composizione adeguata di DeepONets e FNOs per approssimare mappe da operatori a funzioni. Vengono presentati una varietà di esperimenti per dimostrare che NIO supera significativamente le *baseline* e risolve i problemi inversi delle EDP in modo robusto e accurato. Inoltre, NIO è diversi ordini di grandezza più veloce rispetto ai metodi tradizionali per la risoluzione dei problemi inversi.

Contents

1 Introduction **11**
1.1 Outline . . . 14

2 Preliminaries **15**
2.1 Abstract Partial Differential Equation . . . 15
2.2 Forward Problems for Partial Differential Equations . . . 16
2.3 Inverse Problems for Partial Differential Equations . . . 16
2.3.1 Parameter Identification . . . 17
2.3.2 Data Assimilation . . . 18
2.4 Quadrature . . . 18
2.5 Artificial Neural Networks . . . 19
2.5.1 Feedforward Dense Neural Networks . . . 19
2.5.2 Convolutional Neural Networks . . . 21

3 Physics Informed Neural Networks for the Forward Problem of PDEs **25**
3.1 General Description of Physics Informed Neural Networks . . . 25
3.2 Abstract Estimate of the Generalization Error . . . 26
3.3 Semi-linear Parabolic equations . . . 29
3.3.1 The underlying PDEs . . . 29
3.3.2 PINNs . . . 29
3.3.3 Estimate on the generalization error. . . . 31
3.4 Viscous scalar conservation laws . . . 33
3.4.1 The underlying PDE . . . 33
3.4.2 PINNs . . . 34
3.4.3 Estimate on the generalization error. . . . 35
3.5 Incompressible Euler Equations . . . 38
3.5.1 The underlying PDE . . . 38
3.5.2 PINNs . . . 39
3.5.3 Estimate on the generalization error. . . . 40
3.6 Korteweg de-Vries and Kawahara equations . . . 42
3.6.1 The underlying PDEs . . . 43
3.6.2 PINNs . . . 43
3.6.3 Estimate on the generalization error . . . 45
3.7 Numerical Experiments . . . 46
3.7.1 Ensemble Training . . . 46
3.7.2 Semi-linear Parabolic Equation . . . 46
3.7.3 Viscous Scalar Conservation Law . . . 49
3.7.4 Incompressible Euler Equations . . . 51
3.7.5 KdV equation . . . 55
3.7.6 Kawahara equation . . . 58

4 wPINNs for the Forward Problem of Hyperbolic Conservation Laws 61
4.1 Scalar Conservation Laws . . . 61
4.2 Weak PINNs (*wPINNs*) . . . 62
4.3 Estimate of the Generalization Error . . . 64
4.4 Implementation of *wPINNs* . . . 66
4.4.1 Ensemble Training . . . 66
4.4.2 Random Reinitialization of the Test function Parameters . . . 66
4.4.3 Averages of retrainings . . . 66
4.5 Numerical Experiments . . . 67
4.5.1 Standing and Moving Shock . . . 68
4.5.2 Rarefaction Wave . . . 70
4.5.3 Sine Wave Initial Datum . . . 72

5 Physics Informed Neural Networks for the Inverse Problem of PDEs 73
5.1 PINNs for the Parameter Identification Problem . . . 73
5.2 PINNs for the Data Assimilation Problem . . . 74
5.2.1 An abstract estimate on the generalization error . . . 76
5.3 Poisson's equation . . . 79
5.3.1 The underlying inverse problem . . . 79
5.3.2 PINNs . . . 80
5.3.3 Estimates on the generalization error . . . 81
5.4 Heat Equation . . . 82
5.4.1 The underlying inverse problem . . . 82
5.4.2 PINNs . . . 83
5.4.3 Estimates on the generalization error . . . 84
5.5 The Wave equation . . . 85
5.5.1 The underlying inverse problem . . . 85
5.5.2 PINNs . . . 86
5.5.3 Estimates on the generalization error . . . 87
5.6 The Stokes equation . . . 88
5.6.1 The underlying inverse problem . . . 89
5.6.2 PINNs . . . 90
5.6.3 Estimates on the generalization error . . . 91
5.7 Numerical experiments . . . 93
5.7.1 Poisson's Equation . . . 93
5.7.2 Heat equation . . . 95
5.7.3 Wave Equation . . . 97
5.7.4 Stokes Equation . . . 99

6 Physics Informed Neural Networks for High-Dimensional PDEs 103
6.1 A Motivating Example . . . 103
6.2 Radiative Transfer Equation . . . 105
6.2.1 The underlying PDEs . . . 107
6.2.2 PINNs . . . 107
6.2.3 Estimates on the generalization error . . . 108
6.2.4 Numerical Experiments . . . 109
6.3 Kolmogorov Equations . . . 119
6.3.1 PINNs . . . 120
6.3.2 Numerical Experiments . . . 121

7 Operator Learning **125**
7.1 DeepONet . . . 126
7.2 Neural Operators . . . 127
7.2.1 Fourier Neural Operators . . . 128
7.3 Neural Inverse Operators for solving PDE Inverse problems . . . 129
7.3.1 Mathematical Framework. . . . 129
7.3.2 Learning Task and Challenges . . . 132
7.3.3 A Motivating (Formal) Calculation. . . . 133
7.3.4 The Architecture. . . . 134
7.4 Numerical Experiments. . . . 136
7.4.1 Calderón Problem for EIT. . . . 137
7.4.2 Inverse Wave Scattering. . . . 139
7.4.3 Radiative Transport Equation and Optical Imaging. . . . 140
7.4.4 Seismic Imaging. . . . 141
7.4.5 Numerical Results . . . 141
7.4.6 Comparison with Standard Numerical Methods for Inverse Problems . . . 154

8 Conclusions **159**

9 Appendix **163**
9.1 Radiative Transfer . . . 163
9.1.1 Estimates on the Generalization Error for the Radiative Transfer Equation . . . 163
9.1.2 Estimates on the Generalization Error in the Steady Case . . . 166
9.2 Neural Inverse Operator . . . 168
9.2.1 Architecture Details . . . 168
9.2.2 Training Details . . . 172

References **1**

1 Introduction

Deep learning has emerged as a central tool in science and technology in the last few years. It is based on using deep artificial neural networks (DNNs), which are formed by composing many layers of affine transformations and scalar non-linearities. These deep neural networks have been applied with tremendous success [1] in a variety of tasks such as image and text classification, speech and natural language processing, robotics, game intelligence, and protein folding [2], among others.

Partial differential equations (PDEs) model a vast array of natural and manmade phenomena in all areas of science and technology. Explicit solution formulas are only available for very specific types and examples of PDEs. Hence, numerical simulations are necessary for most practical applications featuring PDEs. A diverse set of methods for approximating PDEs numerically are available, such as finite difference, finite element, finite volume and spectral methods. Although very successful in practice, it is still challenging to numerically simulate problems such as Uncertainty Quantification (UQ), multi-scale and multi-physics problems, Inverse and constrained optimization problems, PDEs in domains with very complex geometries, and PDEs in very high dimensions.

Deep learning techniques are being increasingly used in the numerical approximations of PDEs. A brief and very incomplete survey of this rapidly emerging literature follows: one approach in using deep neural networks (DNNs) for numerically approximating PDEs is based on explicit (or semi-implicit) representation formulas such as the Feynman-Kac formula for parabolic (and elliptic) PDEs, whose compositional structure is in turn utilized to facilitate approximation by DNNs. This approach is presented and analyzed for a variety of (parametric) elliptic and parabolic PDEs in [3, 4, 5] and references therein, see a recent paper [6] for a similar approach to approximating linear transport equations with deep neural networks.

Another strategy is to enhance existing numerical methods by adding deep learning inspired modules into them. The paradigm has become popular with the name of *differentiable physics*. On this line of thought several works have been proposed. In [7, 8] it was proposed to learn free parameters of numerical schemes from data. In [9], the authors show that neural networks used *in the loop* of differentiable simulators can improve the solution accuracy of the solver by correcting for effects not captured by the discretized PDE.

Finally, deep learning has found a large application in scientific computing particularly for the construction of *fast surrogates* for approximating a variety of *high-dimensional* partial differential equations (PDEs) including elliptic [10, 11], parabolic [4] and hyperbolic PDEs [12, 13].

A prominent category of high-dimensional problems is known as *many-query problems*, where the high-dimensionality arises from the large *parameters space* involved. For instance in the context of aero-dynamic, a fundamental task is the optimization of the shape of an airfoil in order to maximize its aerodynamic efficiency. A standard practice in this field is to perturb a reference shape with a given class of parametric functions, which depends on several parameters (of order of hundreds), and then maximize the efficiency with respect to such set of parameters. In order to do that, the underlying partial differential equation, corresponding in this case to the well-know Euler equations, is solved for a very large number of instances, each corresponding to a particular realization of the input parameter space. Clearly, querying the computationally costly PDE solver multiple times renders these problems prohibitively expensive. Many-query problems extend beyond aerodynamics and PDE-constrained optimization, and encompass a broad range of problems in scientific computing, including uncertainty quantification, Bayesian inversion, among others.

One additional class of high-dimensional partial differential equations (PDEs) includes problems characterized by a inherently large *state space*. Examples of such PDEs include the Boltzmann equation, Radiative transfer equation, Schrödinger equation, and Black-Scholes equation with large number of spatial dimensions. Conventional grid-based numerical methods, such as finite elements or finite differences, prove inadequate in accurately approximating solutions for these PDEs. This inadequacy arises from the well-established *curse of dimensionality*, whereby the computational cost required to solve these problems to a desired accuracy increases exponentially with the dimensionality of the system. Consequently, these PDEs become computationally intractable using traditional approaches.

Given that neural networks are very powerful universal function approximators [14, 15], it is natural to consider the space of neural networks as an ansatz space for approximating solutions of these PDEs. In particular, neural networks can be used to learn the map from the high-dimensional input space to, either the entire *solution field*, or relevant *observables* (for instance the efficiency of the airfoil in the case of the shape optimization). This is usually done by minimizing a suitable loss function.

Two different philosophies have emerged to build fast surrogates: one, which falls under the rubric of *supervised learning*, aims at collecting sufficient amount of training data, generated from either observations (experiments) or numerical simulations, and minimizing the mismatch between the training data and the model predictions. However, despite a few techniques have been developed over the years to build surrogates even with very small amount of data [16, 12], the generation of the data can nonetheless still be very expensive. Such approach fits very well the class of *many-query* problems, which have intrinsic low dimensionality of the state space (up to 4 dimensions). However, in many contexts, for instance in the case of inherently high-dimensional PDEs one needs a learning framework that can approximate the underlying PDE solutions with possibly *no data*. The second philosophy is founded on *physics informed loss* and corresponding physics informed neural networks (PINNs), which provide a very popular example of such an *unsupervised learning* framework. In contrast to supervised learning approaches where the mismatch between the ground truth and neural network predictions is minimized during training, training of PINNs relies on the minimization of an underlying (pointwise) residual associated with the PDE in suitable hypotheses spaces of neural networks.

There has been an explosive growth of papers that present algorithms with PINNs for various applications to both forward and inverse problems for PDEs and a very incomplete list of references include [17, 18, 19, 20, 21] and references therein. Needless to say, PINNs have emerged as a very successful paradigm for approximating different aspects of solutions of PDEs. However why do PINNs approximate a wide variety of PDEs so well? Although many heuristic reasons have been proposed in some of the afore-cited papers there is very little rigorous justification of why PINNs work. With the exception of the very recent paper [22], there are few rigorous bounds on the approximation error due to PINNs. In [22], the authors prove consistency of PINNs by showing convergence, under reasonable hypothesis, to solutions of linear elliptic and parabolic PDEs as the number of training samples is increased.

The first part of this thesis aims at providing some rationale of why PINNs are so efficient at approximating solutions for the *forward problem* for PDEs, under reasonable and verifiable hypothesis on the underlying PDE. The key issue that we wish to address is *to understand the mechanisms by which minimizing the PDE residuals at collocation points, which is the main ingredient of the PINNs training algorithm, might lead to control (bounds) on the overall error.* To this end, we will present an abstract framework for PINNs that encompasses a wide variety of potential applications, including elliptic and parabolic nonlinear PDEs, and prove estimates on the so-called *generalization error* i.e, the error of the neural network on predicting unseen data.

These elements also bring forth the *limitations of PINNs* by highlighting PDEs where PINNs might not provide an accurate approximation. In particular, there are a large number of contexts in which solutions of PDEs might not be smooth. A prototypical example of a class of PDEs for which the underlying solutions are not smooth, is provided by nonlinear hyperbolic systems of conservation laws such as the Euler, shallow-water and ideal Magneto-Hydrodynamics (MHD) equations [23]. Even in the simplest case of a *scalar conservation law*, e.g. inviscid Burgers' equation, it is well-known that, even if the initial datum is smooth, discontinuities in the form of *shock waves* form within a finite time. Hence, the underlying equation can no longer be interpreted in a (pointwise) *strong* sense, rather *weak solutions* need to be considered [23]. Moreover, these weak solutions are no longer unique and additional admissibility criteria or *entropy conditions* have to be imposed in order to restore uniqueness [23]. Given this context, we ask if one can modify PINNs to design an *unsupervised learning framework* for approximating (entropy) weak solutions of hyperbolic conservation laws. To this end, we will focus on the simple yet prototypical case of scalar conservation laws here and propose a suitable *weak* versions, named *wPINNs*.

PINNs have also emerged very succesfully as an attractive framework for approximating the so-called PDEs *inverse problems* (see [24, 17, 19, 25] and references therein). For such inverse problems, one does not necessarily have complete information on the inputs to the PDEs, such as initial data, boundary conditions and coefficients. Hence, the so-called *forward problem* cannot be solved uniquely. However, we have access to (noisy) data for (observables) of the underlying solution. The aim is to use this data in order to determine the unknown inputs of the PDEs and consequently its solution. In this context, data and physics informed loss can be combined in order to efficiently solve inverse problems. However, also in this case, very few rigorous results on the approximation error with PINNs are available. The goal of the third part of this work is to *derive rigorous bounds on the generalization errors of PINNs for approximating solutions to a class of inverse problems for PDEs.* We will focus on the so-called *unique continuation* or *data assimilation* problems, where boundary conditions to the PDEs are unknown and need to be inferred from (possibly noisy) measurements of certain observables of the underlying solution.

We will conclude the first part of the thesis, by focusing on the second class of high-dimensional PDEs, which are distinguished by their large state space. Specifically, we will employ a physics-informed neural network (PINN) to approximate the solutions of both the Radiative Transfer equation and the Black-Scholes equation in extremely high-dimensional settings. By doing so, we aim to illustrate the advantage of utilizing the PINN loss function, thereby substantiating the initial assertion regarding the relevance of PINNs in the context of scientific computing.

In the last part of the thesis, we will focus on *operator learning*, which has very recently emerged as a dominant paradigm in the applications of machine learning to PDEs. As a matter of fact, constructing fast surrogates to specifically solve *many-query* problems boils down to approximate the solution *operator* underlying a given partial differential equation. As the task at hand in this context is to learn the underlying solution operator, recently developed *operator learning* methods can be employed in this infinite-dimensional setting. A very partial list of architectures for operator learning include operator networks [26], DeepONets [27] and its variants [28, 29, 30], PCA-net [31] , neural operators [32] such as graph neural operator [33], Multipole neural operator [34] and the very popular Fourier Neural Operator [35] and its variants [36, 37], VIDON [38], spectral neural operator [39], LOCA [40], NOMAD [41], transformer based operator learning architectures [42], and the very recent convolutions-based neural operator CNO [43]. Moreover, as in the case of neural networks, also for operator learning architectures a data or physics-inspired loss can be employed, see [44] and references therein.

Given the widespread success of operator learning and other deep learning-based algorithms in the context of forward problems for PDEs, it is natural to investigate their utility in learning the solutions of the corresponding inverse problems from *data*. A large class of such inverse problems takes the following abstract form: given observables as operators (mappings between function spaces), infer the underlying input coefficient (functions) of the associated PDE. A prototypical example is the well-studied *Calderón Problem* [45] that arises in electrical impedance tomography (EIT) in medical imaging. Here, the observable is the Dirichlet-to-Neumann (DtN) operator that maps the voltage on the boundary to the current, and one is interested in inferring the underlying conductivity field. A related example is inverse wave scattering for geophysical applications. Other examples include optical tomography [46] where the observable is the so-called *Albedo operator*, and one needs to infer the scattering and absorption coefficients of the underlying medium. Another prominent example arises in *seismic imaging* in geophysics [47] where the observable is the source-to-receiver (StR) operator, and the task at hand is to infer the underlying sub-surface properties such as wave velocity or material density. In many of these examples, the solution to the resulting inverse problem is unique and stable if and only if the inverse problem is posed as a *mapping from operators to functions*. Hence, one needs non-trivial modifications of existing operator learning architectures to handle inverse problems. We conclude by proposing a novel architecture, termed as *Neural Inverse Operators* (NIOs), for learning solutions of these inverse problems from data and test it extensively on a suite of problems, including the Calderón problem in EIT, inverse wave scattering for object detection, reconstructing the absorption and scattering coefficients in optical tomography, and seismic wave migration.

1.1 Outline

For the convenience of the reader, we provide a brief summary of the contents of the present thesis:

- Chapter 2 provides an overview of the preliminaries necessary for the entire thesis. We establish an abstract framework for forward and inverse problems involving partial differential equations (PDEs) and introduce two relevant architectures used in deep learning: feed-forward and convolutional neural networks.
- In Chapter 3, we formulate physics-informed neural networks (PINNs) for the abstract forward problem and prove an estimate on the generalization error. Subsequently, we apply this abstract framework and error estimate to concrete examples, including semi-linear Parabolic PDEs, viscous scalar conservation laws, incompressible Euler equations of fluid dynamics, and dispersive equations.
- Chapter 4 focuses on the weak formulation of PINNs and presents the solution to the forward problem for an extensive set of experiments based on scalar conservation laws.
- In Chapter 5, we describe the PINNs algorithm for parameter identification and unique continuation (data assimilation) problems for abstract PDEs and derive an estimates on the generalization error for the latter class. We will then prove specific estimates for the Poisson, Heat, Wave and Stokes equations.
- Concluding the first part of the thesis that revolves around PINNs, Chapter 6 delves into high-dimensional state space PDEs, particularly Radiative Transfer and Black-Scholes equations.
- Finally, in Chapter 7 we describe the general framework of operator learning and present Neural Inverse Operator to approximate the inverse map underlying a large class of PDEs inverse problems.

2 Preliminaries

In this chapter, we outline the essential elements upon which the entire thesis is built on. We begin by providing a comprehensive formulation of the forward and inverse problem for Partial Differential Equations (PDEs). Specifically, we address two distinct classes of inverse problems: (1) parameter identification and (2) data continuation problems. The chapter concludes with the description of two relevant network architectures: *feed-forward* and *convolutional* neural networks.

2.1 Abstract Partial Differential Equation

Let $D_T \subset \mathbb{R}^{\bar{d}}$, for some $\bar{d} \geq 1$, be the underlying domain, with smooth (C^1) boundary, denoted by ∂D_T. We include space-time domains by setting $D_T = D \times (0,T)$, $D \subset \mathbb{R}^d$ with $d \geq 1$. In this case $\bar{d} = d + 1$ and the *space-time boundary* is $\partial D_T = \partial D \times (0,T) \cup D \times \{t = 0\}$, with ∂D denoting the smooth boundary of D. Let

$$\begin{aligned} \mathcal{U}(D_T) &= L^{p_u}(D_T;\mathbb{R}^m), \\ \mathcal{S}(D_T) &= L^{p_s}(D_T;\mathbb{R}^m) \\ \mathcal{A}(D_T) &= L^{p_a}(D_T;\mathbb{R}^m), \end{aligned} \tag{2.1}$$

be Banach spaces, with

$$L^p(D_T;\mathbb{R}^m) = \left\{ f : D_T \to \mathbb{R}^m, \ s.t \int_{D_T} |f(z)|^p dz < \infty \right\} \tag{2.2}$$

and $m \geq 1$, $1 \leq p_u, p_a, p_s < \infty$. For ease of notation, we omit the explicit dependence of the domain when the domain is known from the context, i.e. $\mathcal{U}(D_T) = \mathcal{U}$.

A generic abstract partial differential equation (PDE) is represented by the following differential equation,

$$\mathcal{D}_a(u) = s. \tag{2.3}$$

Here, $\mathcal{D}_a : \mathcal{U}(D_T) \to \mathcal{S}(D_T)$ is a *differential operator* and $s \in \mathcal{S}(D_T)$ is the *source term*. The differential operator may also depend on some equation coefficient $a \in \mathcal{A}(D_T)$.

Given the definitions above, it follows

$$\begin{aligned} (H1): &\quad \|\mathcal{D}_a(u)\|_{\mathcal{S}} < +\infty, \quad \forall u \in \mathcal{U}, \text{ with } \|u\|_{\mathcal{U}} < +\infty \text{ and } \forall a \in \mathcal{A}, \text{ with } \|a\|_{\mathcal{A}} < +\infty. \\ (H2): &\quad \|s\|_{\mathcal{S}} < +\infty. \end{aligned} \tag{2.4}$$

A prototypical example of PDE is the heat equation. Let $t \in [0,T]$ and $x \in D$, then the differential operator becomes:

$$\mathcal{D}_a(u) = \partial_t u - a(x)\Delta_x u \tag{2.5}$$

with $\Delta_x = \sum_{i=1}^{d} \partial_{x_i x_i} u$ being the *Laplacian* of the solution u. The heat equation describes the time-space evolution of the temperature u. The coefficient a in this case represents the *thermal diffusivity*.

2.2 Forward Problems for Partial Differential Equations

Let us further consider the Banach spaces

$$\mathcal{B}(\partial D_T) = L^{p_b}(\partial D_T; \mathbb{R}^m), \quad \mathcal{B}_u(\partial D_T) = L^{p_{u_b}}(\partial D_T; \mathbb{R}^m), \tag{2.6}$$

and the generic boundary operator $\mathcal{B} : \mathcal{B}_u(\partial D_T) \to \mathcal{B}(\partial D_T)$

$$\mathcal{B}\left(\mathcal{T}u\right) = f_b, \tag{2.7}$$

with $\mathcal{T} : \mathcal{U}(D_T) \to \mathcal{B}_u(\partial D_T)$ being the *trace operator* and $f_b \in \mathcal{B}(\partial D_T)$. We note that the generic boundary condition also includes initial conditions when $D_T = (0,T) \times D$. From the definitions above it follows

$$\begin{aligned} &(H3): \quad \|\mathcal{B}\left(\mathcal{T}u\right)\|_{\mathcal{B}} < +\infty, \quad \forall u \in \mathcal{U}, \text{ with } \|u\|_{\mathcal{U}} < +\infty. \\ &(H4): \quad \|f_b\|_{\mathcal{B}} < +\infty. \end{aligned} \tag{2.8}$$

The *forward problem* for 2.3 consists in finding $u \in \mathcal{U}$, $u : D_T \to \mathbb{R}^m$, such that:

$$\begin{aligned} \mathcal{D}_a(u) &= s, & x \in D_T \\ \mathcal{B}\left(\mathcal{T}u\right) &= f_b, & y \in \partial D_T \end{aligned} \tag{2.9}$$

The forward problem is well-posed if the following three conditions are satisfied:

1. *Existence.* There exist a $u \in \mathcal{U}(D_T)$ such that
$$\begin{aligned} \mathcal{D}_a(u) &= s, & x \in D_T \\ \mathcal{B}\left(\mathcal{T}u\right) &= f_b, & y \in \partial D_T \end{aligned}$$
2. *Uniqueness.* $\forall u, v \in \mathcal{U}(D_T)$, $u = v$ if and only if
$$\begin{aligned} \mathcal{D}_a(v) &= \mathcal{D}_a(v), & x \in D_T \\ \mathcal{B}\left(\mathcal{T}u\right) &= \mathcal{B}\left(\mathcal{T}v\right), & y \in \partial D_T \end{aligned}$$
3. *Stability.* There exist constants $C_{(bd)} = C_{(bd)}(\mathcal{T}u, \mathcal{T}v)$ and $C_{(pde)} = C_{(pde)}(u,v)$ such that $\forall u, v \in \mathcal{U}(D_T)$,
$$\|u - v\|_{\mathcal{U}(D_T)} < C_{(bc)}\omega_{(bc)}\left(\|\mathcal{B}\left(\mathcal{T}u\right) - \mathcal{B}\left(\mathcal{T}v\right)\|_{\mathcal{B}_u(\partial D_T)}\right) + C_{(pde)}\omega_{(pde)}\left(\|\mathcal{D}_a(u) - \mathcal{D}_a(v)\|_{\mathcal{S}(D_T)}\right) \tag{2.10}$$
with $\omega_{(\cdot)} : \mathbb{R}_+ \to \mathbb{R}_+$, denoted as *modulus of continuity*, being a monotonically increasing function with $\lim_{y\to 0}\omega(y) = 0$..

2.3 Inverse Problems for Partial Differential Equations

Inverse problems are ubiquitous across many fields of science and engineering and they can be of different types and nature. In inverse problems, one has usually access to (possibly noisy) measurements of the underlying solution u in the domain D'_T i.e,

$$\mathcal{L}(u) = u', \tag{2.11}$$

with the *observation operator* $\mathcal{L} : \mathcal{U}(D_T) \to \mathcal{Z}(D'_T)$ and *data* $u' \in \mathcal{Z}(D'_T)$. The domain D'_T is usually named *observation domain.* Depending on the specific problem and measurement setup, the observation operator may have different forms. For instance, it could be the identity operator, a restriction operator to a subset of the domain, a trace operator to boundary measurements, etc. As a consequence, the observation domain may correspond to the entire domain D_T, to an its subset, for instance the boundaries of D_T, etc. In general, given the measurements data u' and a state of interest $w \in \mathcal{W}(D_T)$, there exists a forward operator $\mathcal{F} : \mathcal{W}(D_T) \to \mathcal{Z}(D'_T)$, such that $\mathcal{F}(w) = u'$. In the case of partial differential equation, the forward operator arises from (2.9) which links together the state w and the observable u'. Hence, an inverse problem can be generally formulates as: *find* $w \in \mathcal{W}(D_T)$ *such that* $\mathcal{F}(w) = u'$.

Similarly to the forward problem, the inverse one is well-posed if the following three conditions are satisfied:

1. *Existence.* There exists $w \in \mathcal{W}(D_T)$ such that $\mathcal{F}(w) = u'$

2. *Uniqueness.* $\forall w, w' \in \mathcal{W}(D_T)$, $w = w'$ if and only if $\mathcal{F}(w) = \mathcal{F}(w')$

3. *Stability.* There exists a constant $C = C(w, w')$, such that $\forall w, w' \in \mathcal{W}(D_T)$,

$$\|w - w'\|_{\mathcal{W}(D_T)} < C\omega\left(\|\mathcal{F}(u) - \mathcal{F}(v)\|_{\mathcal{U}(D'_T)}\right) \tag{2.12}$$

   with $\omega : \mathbb{R}_+ \to \mathbb{R}_+$, denoted as *modulus of continuity*, being a monotonically increasing function with $\lim_{y \to 0} \omega(y) = 0$.

In the rest of this work, we will focus on two different classes of inverse problems related to partial differential equations, (1) *parameter identification*, and (2) *data assimilation* or *unique continuation problem.*

2.3.1 Parameter Identification

The parameter identification problem consists in finding $a \in \mathcal{A}$ given the data u'. Formally, let us define the *parameter-to-solution map* $\phi : \mathcal{A} \to \mathcal{U}$, $\phi : a \mapsto u$. Since a and u are coupled through the partial differential equation 2.3, the existence of such operator is guaranteed by the implicit function theorem under reasonable assumptions. Then, we can define the forward operator as

$$\mathcal{F} := \mathcal{L} \circ \phi : \mathcal{A}(D_T) \to \mathcal{Z}(D'_T). \tag{2.13}$$

The parameter identification boils down to finding $a \in \mathcal{A}$ such that $\mathcal{F}(a) = u'$. In the parameter identification problem the observation operator is usually defined as the identity. However, for some inverse problems, including electrical impedance tomography, inverse scattering, optical imaging, etc., the observation operator is a trace operator to the boundary of D_T. For instance, in electrical impedance tomography (EIT), different electrical voltages are applied at the boundary of an object, and the arising electrical currents are measured there. From these measurements one would like to reconstruct the conductivity as a function of space, which gives information about different materials inside the object. The last class of problems is described in more detail in the next section and in Section 7.3.1.

Impedance Tomography, Inverse Scattering, Optical Imaging, and Sesimic Imaging

A large class of parameter identification problems takes the following abstract form: given an *operator* as observable, infer the underlying input coefficient $a \in \mathcal{A}$ of the associated PDE. Specifically, such operators are *boundary operators*

$$\Lambda_a : \mathcal{B}(\partial D_T) \to \mathcal{H}(\partial D_T), \tag{2.14}$$

which map the boundary data $f_b \in \mathcal{B}(\partial D_T)$ to the measurement $\Lambda_a(f_b) = h(u) \in \mathcal{H}(\partial D_T)$, with $\mathcal{H}$ a Banach space on ∂D_T. In the example of the EIT, for instance, *the collection of all the voltage-current measurements represent a discrete realization of such boundary operator.* Hence, one can rewrite the forward problem associated with the PDE (2.3) as,

$$\mathcal{F} : \mathcal{A}(D_T) \to \mathcal{L}\left(\mathcal{B}(\partial D_T), \mathcal{H}(\partial D_T)\right), \ a \mapsto \mathcal{F}(a) = \Lambda_a, \tag{2.15}$$

where Λ_a is the boundary observation operator (2.14) and $\mathcal{L}(X, Y)$ denotes continuous operators between function spaces X and Y.

The inverse map for the forward problem (2.15) takes the form

$$\mathcal{F}^{-1} : \mathcal{L}\left(\mathcal{B}(\partial D_T), \mathcal{H}(\partial D_T)\right) \to \mathcal{A}(D_T), \ \Lambda_a \mapsto a = \mathcal{F}^{-1}(\Lambda_a), \tag{2.16}$$

The rigorous guarantee of the existence and, more importantly, the uniqueness of this inverse map $\mathcal{F}^{-1}$ for a large class of PDEs, is a crowning achievement of the mathematical theory of inverse problems [48]. Moreover, one can also show Lipschitz or α-Hölder-stability of the inverse problem by proving estimates of the form,

$$\|\mathcal{F}(a) - \mathcal{F}(\bar{a})\|_L \sim \|a - \bar{a}\|_{\mathcal{A}}^{\alpha}, \quad 0 < \alpha \leq 1, \quad \forall a, \bar{a} \in \mathcal{A}(D_T) \tag{2.17}$$

In some cases, the right-hand side of the above stability estimate is replaced by a logarithm of $\|a - \bar{a}\|_{\mathcal{A}}$, which only guarantees (weak) logarithmic stability.

2.3.2 Data Assimilation

Differently from the parameter identification problem, in the data assimilation, we assume that the generic boundary conditions (which include initial conditions) f_b in (2.7) are not known. Moreover, in this case, the observation operator corresponds to a *restriction operator* to the subset $D'_T \subset D_T$. Then, in solving the data assimilation problem, one determines the function $u \in \mathcal{U}(D_T)$ and consequently the boundary conditions $\mathcal{T}(u)$ from the data (2.11), given measurements *only* on the observation domain D'_T. Equivalently, given the non-linear operator

$$\mathcal{F} := \mathcal{L} \circ \phi : \mathcal{B}(\partial D_T) \to \mathcal{Z}(D'_T), \tag{2.18}$$

with $\phi : \mathcal{B} \to \mathcal{U}$, $\phi : f_b \mapsto u$, being the *boundary-to-solution map*, the data assimilation problem consists in finding $u \in \mathcal{U}(D_T)$ and $f_b \in \mathcal{B}(\partial D_T)$ such that $\mathcal{F}(u, f_b) = u'$.

2.4 Quadrature

In the following section, we consider approximating integrals of functions with quadrature. To this end, we consider a mapping $g : D_T \to \mathbb{R}^m$, with g living in a suitable Banach space. We are interested in approximating the integral,

$$\overline{g} := \int\limits_{D_T} g(y) dy,$$

with dy denoting the $\bar{d}$-dimensional Lebesgue measure. In order to approximate the above integral by a quadrature rule, we need the quadrature points $y_i \in D_T$ for $1 \leq i \leq N$, for some $N \in \mathbb{N}$ as well as weights w_i, with $w_i \in \mathbb{R}_+$. Then a quadrature is defined by,

$$\overline{g}_N := \sum_{i=1}^{N} w_i g(y_i), \tag{2.19}$$

for weights w_i and quadrature points y_i. We further assume that the quadrature error is bounded as,

$$|\overline{g} - \overline{g}_N| \leq C_{quad}\left(\|g\|_{Z^*}, \bar{d}\right) N^{-\alpha}, \tag{2.20}$$

for some $\alpha > 0$.

As long as the domain D_T is in reasonably low dimension i.e $\bar{d} \leq 4$, we can use standard (composite) quadrature rules on an underlying grid. In this case, the quadrature points and weights depend on the order of the quadrature rule [49] and the rate α depends on the regularity of the underlying integrand i.e, on the space Z^*. As an example let us consider the midpoint rule. For $M \in \mathbb{N}$, we partition $D_T \subset \mathbb{R}^{\bar{d}}$ into $N \sim M^{\bar{d}}$ cubes of edge length $\frac{1}{M}$ and we denote by $\mathcal{S} = \{y_n\}_{n=1}^{N}$ the midpoints of these cubes. The formula and accuracy of the midpoint rule are then given by,

$$\overline{g}_N := \frac{1}{N}\sum_{i=1}^{N} g(y_i), \quad |\overline{g} - \overline{g}_N| \leq \left(\frac{1}{24}\sum_{i=1}^{\bar{d}} \frac{\partial^2 g}{\partial x_i^2}\right) N^{-\frac{2}{\bar{d}}}. \tag{2.21}$$

Because of the exponential dependence of the error on the dimension d of the underlying domain D_T, these grid based quadrature rules are not suitable for domains in high dimensions. For moderately high dimensions i.e $4 \leq \bar{d} \leq 20$, we can use *low discrepancy sequences*, such as the Sobol and Halton sequences, as quadrature points [50]. As long as the integrand g is of bounded *Hardy-Krause variation* [51], the error in (2.20) converges at a rate $(\log(N))^{\bar{d}} N^{-1}$,

$$\overline{g}_N := \frac{1}{N}\sum_{i=1}^{N} g(y_i), \quad |\overline{g} - \overline{g}_N| \leq V_{HK}(g)\,\frac{(\log(N))^{\bar{d}}}{N} \tag{2.22}$$

Here, V_{HK} denotes the Hardy-Krause variation and $V_{HK}(g) \leq \infty$

For problems in very high dimensions $\bar{d} \gg 20$, Monte-Carlo quadrature is the numerical integration method of choice [50]. In this case, the quadrature points are randomly chosen, independent and identically distributed (with respect to a scaled Lebesgue measure). Given the non-deterministic nature of the quadrature, the quadrature error has to be defined in a probabilistic sense,

$$\overline{g}_N := \frac{1}{N}\sum_{i=1}^{N} g(y_i), \quad \int_{D_T^L} |\overline{g} - \overline{g}_N|^2 \, d\mu^L(\mathcal{S}) \leq \frac{\mathbb{V}(g)^{\frac{1}{2}}}{N^{\frac{1}{2}}} \tag{2.23}$$

with $\mu^L(\mathcal{S})$ being the induced product measure on the quadrature points $\mathcal{S}$ and $\mathbb{V}(g)$ the variance of g.

2.5 Artificial Neural Networks

2.5.1 Feedforward Dense Neural Networks

Given an input $x \in D_T$, a feedforward neural network, also termed as a Multi-Layer Perceptron (MLP), shown in figure 2.1, transforms it to an output, through a layer of units (neurons) which compose of either affine-linear maps between units (in successive layers) or scalar non-linear activation functions within units [52], resulting in the representation,

$$u_\theta(x) = C_K \circ \sigma \circ C_{K-1} \ldots\ldots\ldots \circ \sigma \circ C_2 \circ \sigma \circ C_1(x). \tag{2.24}$$

Here, $\circ$ refers to the composition of functions and σ is a scalar (non-linear) activation function. A large variety of activation functions have been considered in the machine learning literature [52]. Popular choices for the activation function σ in (2.24) include the sigmoid function, the hyperbolic tangent function and the $ReLU$ function.

For any $1 \leq k \leq K$, we define

$$C_k z_k = W_k z_k + b_k, \quad \text{for } W_k \in \mathbb{R}^{d_{k+1} \times d_k}, z_k \in \mathbb{R}^{d_k}, b_k \in \mathbb{R}^{d_{k+1}}, \tag{2.25}$$

For consistency of notation, we set $d_1 = \bar{d}$ and $d_K = m$.

Thus in the terminology of machine learning (see also figure 2.1), our neural network (2.24) consists of an input layer, an output layer and $(K-1)$ hidden layers for some $1 < K \in \mathbb{N}$. The k-th hidden layer (with d_k neurons) is given an input vector $z_k \in \mathbb{R}^{d_k}$ and transforms it first by an affine linear map C_k (2.25) and then by a nonlinear (component wise) activation σ. A straightforward addition shows that our network contains $\left(\bar{d} + m + \sum\limits_{k=2}^{K-1} d_k\right)$ neurons. We also denote,

$$\theta = \{W_k, b_k\}, \; \theta_W = \{W_k\} \quad \forall\, 1 \leq k \leq K, \tag{2.26}$$

to be the concatenated set of (tunable) weights for our network. It is straightforward to check that $\theta \in \Theta \subset \mathbb{R}^M$ with

$$M = \sum_{k=1}^{K-1} (d_k + 1) d_{k+1}. \tag{2.27}$$

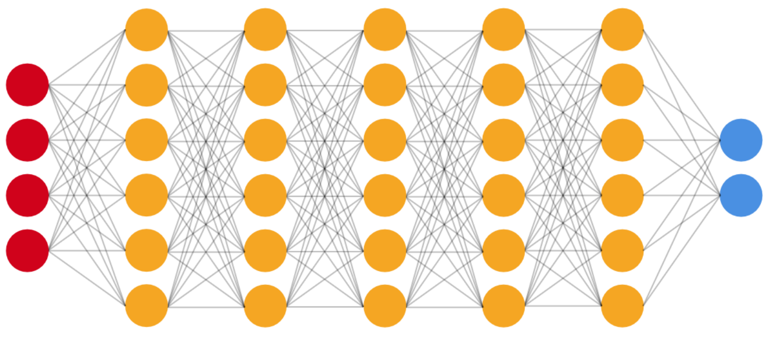


Figure 2.1: An illustration of a (fully connected) deep neural network. The red neurons represent the inputs to the network and the blue neurons denote the output layer. They are connected by hidden layers with yellow neurons. Each hidden unit (neuron) is connected by affine linear maps between units in different layers and then with nonlinear (scalar) activation functions within units.

2.5.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specialized type of Multi-Layer Perceptrons (MLPs) [52]. CNNs are widely employed in image processing tasks such as image classification, segmentation, and video recognition. They have also found applications in time series analysis and natural language processing. The fundamental building blocks of CNNs are the *convolution operation* and *convolutional layers*. In the following sections, we will describe the relevant operations in convolutional neural networks. Finally, we will discuss a specific type of architecture known as Fully Convolutional Neural Networks (FCNNs), which solely consist of convolutional layers.

Convolution Operation

Let's consider $x \in \mathbb{R}^{h \times w \times c}$, where h, w, and c denote the width, height, and number of channels of x. Let $K_w \in \mathbb{R}^{k \times k \times c \times \hat{c}}$ be a discrete kernel of size k.

The discrete, multi-channel convolution with an s-stride of the input x is defined as follows:

$$\mathcal{K} : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^{\hat{h} \times \hat{w} \times \hat{c}}, \tag{2.28}$$

$$\mathcal{K}(x) = (x \star K_w)[i, j, \hat{l}] := \sum_{m,n=0}^{k-1} \sum_{l=1}^{c} K_w[m, n, l, \hat{l}] \cdot x[is + m, js + n, l], \tag{2.29}$$

where l, $\hat{l}$ correspond to the indices of the input and output channels, respectively, and $i = 0, \ldots, h - 1$, $j = 0, \ldots, w - 1$, $\hat{l} = 1, \ldots, \hat{c}$. To mitigate the loss of information at the borders of the input feature map, padding of size p can be performed. This involves adding extra pixels around the border of the input feature map before convolution. Hence, a convolution operator can be characterized by the hyperparameters $(k, s, p, c, \hat{c})$. The kernel size, padding, and stride can be chosen appropriately to achieve a downsampling of the input map with a desired scaling. Figure 2.2 provides a schematic representation of the convolution.

Transposed Convolution Operation

Let's define $y \in \mathbb{R}^{\hat{h} \times \hat{w} \times \hat{c}}$, where $\hat{h}$, $\hat{w}$, and $\hat{c}$ represent the height, width, and number of channels of y. A transposed convolution, denoted as $\mathcal{K}^T(y)$, with a kernel K_w and stride s, transforms y into the output $x \in \mathbb{R}^{h \times w \times c}$, where $h = (s \cdot (\hat{h} - 1) + k)$ and $w = (s \cdot (\hat{w} - 1) + k)$:

$$\mathcal{K}^T : \mathbb{R}^{\hat{h} \times \hat{w} \times \hat{c}} \to \mathbb{R}^{h \times w \times c}. \tag{2.30}$$

Similar to the convolution operation, a transposed convolution can also have a suitable padding, which in this case involves cropping the output x. A schematic representation of a transposed convolution is shown in Figures 2.2b and 2.2c. Just like with convolutions, a transposed convolution can be characterized via the hyperparameters $(k, s, p, c, \hat{c})$.

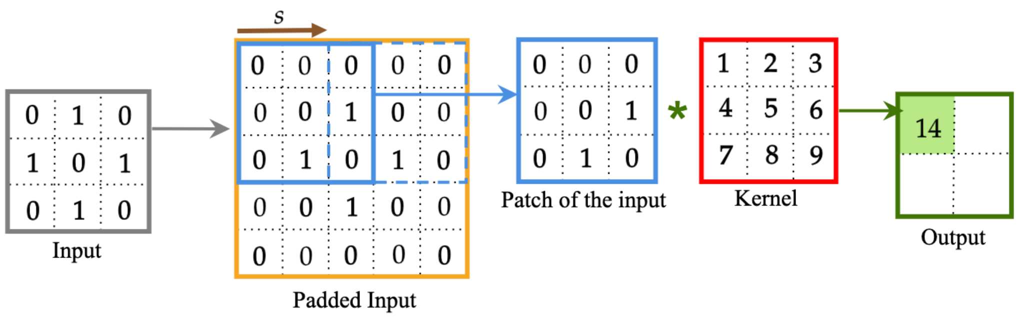
Fully Convolutional Neural Network

Fully Convolutional Neural Networks (FCNNs) are a special class of convolutional networks that can be evaluated for input of any resolution. These networks are composed of an encoder and a decoder, both defined by a series of linear and nonlinear transformations:

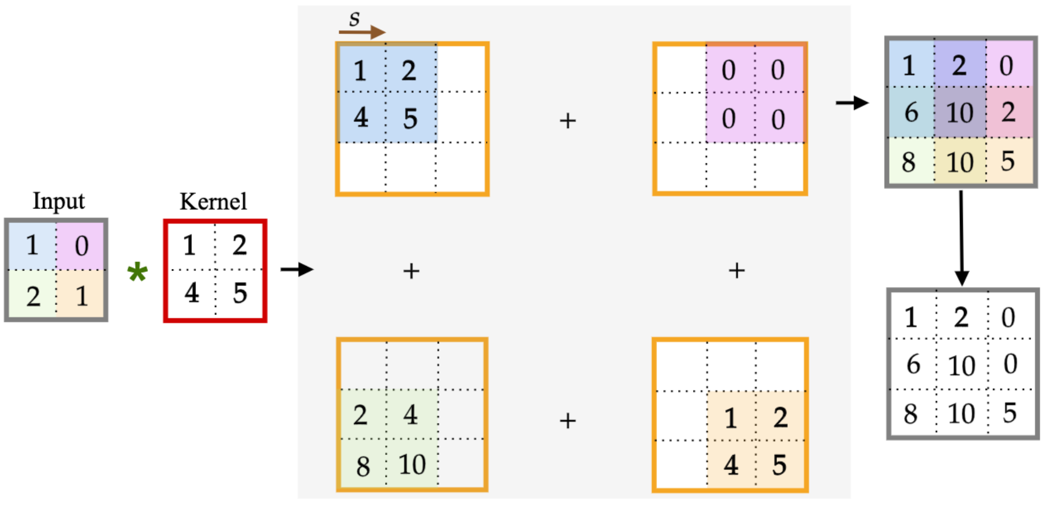
$$\begin{aligned} E_{\theta_e}(y) &= C_L \circ \sigma \circ C_{L-1} \circ \ldots \circ \sigma \circ C_2 \circ \sigma \circ C_1(y), \\ D_{\theta_d}(z) &= C_L^T \circ \sigma \circ C_{L-1}^T \circ \ldots \circ \sigma \circ C_2^T \circ \sigma \circ C_1^T(z), \\ u_\theta(y) &= D_{\theta_d} \circ E_{\theta_e}(y). \end{aligned} \tag{2.31}$$

The affine transformations C_ℓ and C_ℓ^T typically correspond to the convolution operation in the encoder and the deconvolution operation in the decoder, respectively. The convolution usually performs, with an abuse of notation, a "downsampling" of the signal, whereas the deconvolution an "upsampling". In the 2D case, for instance, as ℓ increases in the encoder, the width and height of the signal progressively decrease, while the number of channels c_ℓ is usually chosen to increase. Conversely, in the decoder, the width and height increase while the number of channels decreases. Examples of fully convolutional models include U-Net, ResNet and CNO [53, 54, 43].

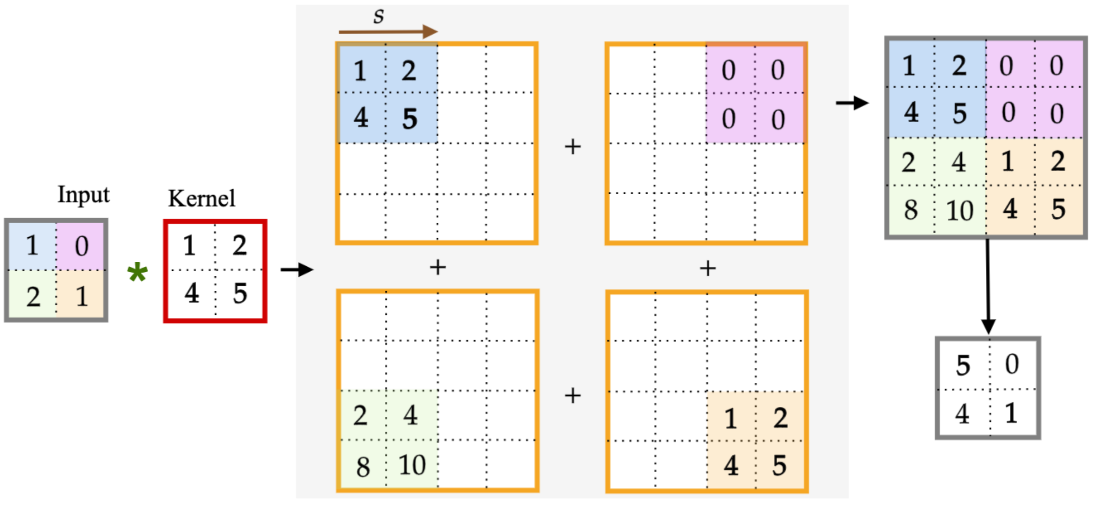
In this thesis, we will focus on classes of FCNNs where the decoder consists of transposed convolutions as described earlier.



(a) Convolution with stride $s = 2$, padding $p = 1$ and kernel K_w of size $k = 3$



(b) Transposed convolution with stride $s = 1$, padding $p = 0$ and kernel K_w of size $k = 2$



(c) Transposed convolution with stride $s = 2$, padding $p = 1$ and kernel K_w of size $k = 2$

Figure 2.2: Schematic representation of Convolution and Transpose Convolution Operators

3 Physics Informed Neural Networks for the Forward Problem of PDEs

In this chapter we provide a general description of physics informed neural networks (PINNs) for approximating the forward problems of partial differential equations. PINNs are first presented in an abstract framework which encompasses a wide variety of potential applications. For this abstract PDE, we provide a bound on the *generalization error*, which identifies possible mechanisms by which PINNs are able to approximate PDEs so well and provide a firm mathematical foundation for approximations by PINNs. Eventually, the abstract error estimate is described in concrete terms for a wide range of specific PDEs, including linear and semi-linear parabolic equations, one-dimensional scalar quasilinear parabolic (and hyperbolic) conservation laws and the incompressible Euler equations of fluid dynamics. In the rest of the thesis, with an abuse of notation, we will denote $|v|^p = \sum_{i=1}^m |v_i|^p$, for all vectors $v \in \mathbb{R}^m$.

3.1 General Description of Physics Informed Neural Networks

Let us consider a feedforward dense neural network u_θ (2.24) with tuning parameters $\theta \in \Theta$. We wish to find the tuning parameters θ such that the resulting neural network u_θ approximate the solution of the forward problem (2.9):

$$u_\theta(x) \approx u(x), \quad x \in D_T \tag{3.1}$$

In order to do so, we define the following residuals:

$$\begin{aligned} &\textit{PDE (or Interior) residual} \qquad && r_{int}[u_\theta] := \mathcal{D}_a(u_\theta) - s, \quad x \in D_T \\ &\textit{Boundary residual} && r_{bc}[u_\theta] := \mathcal{B}(\mathcal{T}u_\theta) - f_b, \quad x \in \partial D_T \end{aligned} \tag{3.2}$$

and corresponding losses,

$$J_{int}(\theta) := \|r_{int}[u_\theta]\|^{p_s}_{\mathcal{S}(D_T)}, \quad J_{bc}(\theta) := \|r_{bc}[u_\theta]\|^{p_b}_{\mathcal{B}(\partial D_T)}. \tag{3.3}$$

By assumptions (H1),(H2) in (2.4), and (H3), (H4) in (2.8), we see that $r_{int}[u_\theta] \in \mathcal{S}$, $\|r_{int}[u_\theta]\|_{\mathcal{S}} < +\infty$, and $r_{bc}[u_\theta] \in \mathcal{B}$, $\|r_{bc}[u_\theta]\|_{\mathcal{B}} < +\infty$ for all $\theta \in \Theta$.

As it will not be possible to evaluate the integral in (3.3) exactly, we need to approximate it numerically by a quadrature rule. To this end, we use the quadrature rules (2.19) discussed earlier and select the *training set* $\mathcal{S}_{int} = \{x_n\}$ with $x_n \in D_T$ for all $1 \leq n \leq N$, and $\mathcal{S}_{bc} = \{y_m, f_b(y_m)\}$, $y_m \in \partial D_T$ for all $1 \leq m \leq M$ as the quadrature points for the quadrature rule (2.19) and consider the following approximation of (3.3):

$$J_{int}(\theta) := \sum_{n=1}^{N} w_n |r_{int}[u_\theta](x_n)|^{p_s}, \quad J_{bc}(\theta) := \sum_{m=1}^{M} w_m |r_{bc}[u_\theta](y_m)|^{p_b} \tag{3.4}$$

Finally, we can minimize the loss function

$$J(\theta) = \lambda J_{int}(\theta) + J_{bc}(\theta), \tag{3.5}$$

with λ being a hyperparameter for balancing the residuals, on account of the PDE and the boundary residuals, over the admissible set of tuning parameters $\theta \in \Theta$ i.e:

$$\theta^* = \underset{\theta \in \Theta}{\arg\min}\, J(\theta), \tag{3.6}$$

and call the resulting neural network $u^* = u_{\theta^*}$ *physics informed neural network (PINN).*

It is common in machine learning [52] to regularize the minimization problem for the loss function i.e we seek to find,

$$\theta^* = \underset{\theta \in \Theta}{\arg\min} \left(J(\theta) + \lambda_{reg} J_{reg}(\theta) \right). \tag{3.7}$$

Here,

$$J_{reg} : \Theta \to \mathbb{R} \tag{3.8}$$

is a *regularization* (penalization) term. A popular choice is to set $J_{reg}(\theta) = \|\theta\|_q^q$ for either $q = 1$ (to induce sparsity) or $q = 2$. The parameter $0 \leq \lambda_{reg} \ll 1$ balances the regularization term with the actual loss J (3.5).

The above minimization problem amounts to finding a minimum of a possibly non-convex function over a subset of $\mathbb{R}^M$ for possibly very large M. We will follow standard practice in machine learning and solve this minimization problem approximately by either (first-order) stochastic gradient descent methods such as ADAM [55] or even higher-order optimization methods such as LBFGS [56].

In summary, a physics informed neural networks is based on the following building blocks:

1. A standard feedforward neural network (as described in section 2.5.1) approximating the solution of the forward problem 2.9, $u_\theta(x) \approx u(x)$, $x \in D_T$
2. Training sets $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{bc}$
3. Interior and boundary residuals $r_{bc}[u_\theta]$, $r_{int}[u_\theta]$
4. A loss function $J(\theta) = \lambda J_{int}(\theta) + J_{bc}(\theta)$

The training algorithm for PINNs is described in Algorithm 1.

3.2 Abstract Estimate of the Generalization Error

In this section, we will estimate the error due to the PINN in approximating the solution u of the forward problem (2.9). The relevant concept of error is the *total error*, often referred to as the so-called *generalization error* (see [57]):

$$\mathcal{E}_G = \mathcal{E}_G(\theta^*; \mathcal{S}_{int}, \mathcal{S}_d) := \|u - u^*\|_{\mathcal{U}(D_T)}. \tag{3.9}$$

Clearly, the generalization error depends on the chosen training set $\mathcal{S}$ and the trained neural network with tuning parameters θ^*, found by minimizing (3.5). However, we will suppress this dependence due to notational convenience.

Algorithm 1: Finding a physics informed neural network to approximate the solution of the forward problem (2.9)

```
Input
    $\mathcal{D}_a$        differential operator on the underlying domain $D_T$
    $s$          input source term
    $f_b$        boundary conditions
    $\mathcal{S}_{int}$       training set for approximating the PDE residual norm
    $\mathcal{S}_{bc}$        training set for approximating the boundary residual norm
    $\overline{\theta}$          initial value of the tunable parameters
    $\lambda$          balancing parameter
Output
    $u_{\theta^*}$       PINN approximating the forward problem 2.9

Step 1: For the initial value of the weight vector $\overline{\theta} \in \Theta$, evaluate the neural network $u_{\overline{\theta}}$ (2.24), the
PDE and boundary residuals (3.2), the loss function (3.5) and its gradients.
Step 2: Run the optimization algorithm until an approximate local minimum $\theta^*$ of (3.5) is reached.
The map $u^* = u_{\theta^*}$ is the desired PINN for approximating the solution $u$ of the forward problem (2.9).
```

Note that there is no computation of the generalization error during the training process. On the other hand, we exclusively monitor the so-called *training error* given by,

$$\mathcal{E}_T := \left((\mathcal{E}_T^{int})^{p_s} + (\mathcal{E}_T^{bc})^{p_b} \right)^{\frac{2}{p_s+p_b}} \tag{3.10}$$

with

$$\mathcal{E}_T^{int} := \left(\sum_{n=1}^{N} w_n |r_{int}[u_{\theta^*}](x_n)|^{p_s} \right)^{\frac{1}{p_s}}, \quad \mathcal{E}_T^{bc} := \left(\sum_{m=1}^{M} w_m |r_{bc}[u_{\theta^*}](y_m)|^{p_b} \right)^{\frac{1}{p_b}}. \tag{3.11}$$

Hence, the training error $\mathcal{E}_T$ can be readily computed, after training has been completed, from the loss function (3.5). The generalization error (3.9) can be estimated in terms of the training error in the following theorem.

Theorem 3.2.1. *Let $u \in \mathcal{U}$ be the unique solution of the forward problem* (2.9) *and assume that the forward problem is stable, namely, for all $u, v \in \mathcal{U}$, it follows*

$$\begin{aligned} \|u - v\|_{\mathcal{U}(D_T)} < C_{bc}\left(\|\mathcal{T}(u)\|_{\mathcal{B}}, \|\mathcal{T}(v)\|_{\mathcal{B}}\right) \|\mathcal{B}\left(\mathcal{T}u\right) - \mathcal{B}\left(\mathcal{T}v\right)\|_{\mathcal{B}(\partial D_T)} + \\ C_{int}(u, v)\|\mathcal{D}_a(u) - \mathcal{D}_a(v)\|_{\mathcal{S}(D_T)} \end{aligned} \tag{3.12}$$

Let $u^* \in \mathcal{U}$ be the PINN generated by (3.6)*, based on the training sets $\mathcal{S}_{int}$, $\mathcal{S}_{bc}$ of quadrature points corresponding to the quadrature rule* (2.19)*. Further assume that the residuals $r_{int}[u_{\theta^*}]$ and $r_{bc}[u_{\theta^*}]$, defined in* (3.2)*, be such that $r_{int}[u_{\theta^*}] \in \mathcal{S}(D_T)$ and $r_{bc}[u_{\theta^*}] \in \mathcal{B}(\partial D_T)$ and the quadrature error satisfies* (2.20)*. Then the following estimate on the generalization error holds,*

$$\mathcal{E}_G \le C_{int}\left(\mathcal{E}_T^{int}\right)^{\frac{p_s}{p_u}} + C_{bc}\left(\mathcal{E}_T^{bc}\right)^{\frac{p_b}{p_u}} + C_{int}\left(C_{quad}^{int}\right)^{\frac{1}{p_u}} N^{-\frac{\alpha_u}{p_u}} + C_{bc}\left(C_{quad}^{bc}\right)^{\frac{1}{p_b}} M^{-\frac{\alpha_b}{p_b}}, \tag{3.13}$$

with constants $C_{int} = C_{int}\left(\|u\|_{\mathcal{U}}, \|u^*\|_{\mathcal{U}}\right)$, $C_{bc} = C_{bc}\left(\|\mathcal{T}(u)\|_{\mathcal{B}}, \|\mathcal{T}(u^*)\|_{\mathcal{B}}\right)$, $C_{quad}^{int} = C_{quad}^{int}\left(\|r_{int}[u_{\theta^*}]\|_{\mathcal{S}}\right)$ and $C_{quad}^{bc} = C_{quad}^{bc}\left(\|r_{bc}[u_{\theta^*}]\|_{\mathcal{B}}\right)$ stemming from (2.10) *and* (2.20)*, respectively. Note that these constants depend on the underlying PINN u^*, which in turn can depend on the number of training points N, M.*

Proof. Let us consider the residuals $r_{int}[u_{\theta^*}]$ and $r_{bc}[u_{\theta^*}]$ corresponding to the trained neural network u^*, defined (3.2). As u solves the forward problem (2.3) we easily see that,

$$r_{int}[u_{\theta^*}] = \mathcal{D}_a(u^*) - \mathcal{D}_a(u), \quad r_{bc}[u_{\theta^*}] = \mathcal{B}\left(\mathcal{T}(u^*)\right) - \mathcal{B}\left(\mathcal{T}(u)\right). \tag{3.14}$$

Hence, we can directly apply the stability bound (3.12) to yield,

$$\begin{aligned}
\mathcal{E}_G &= \|u - u^*\|_{\mathcal{U}} \quad \text{(by (3.9)),} \\
&\leq C_{bc}\|\mathcal{B}\left(\mathcal{T}u\right) - \mathcal{B}\left(\mathcal{T}v\right)\|_{\mathcal{B}(\partial D_T)} + C_{int}\|\mathcal{D}_a(u) - \mathcal{D}_a(v)\|_{\mathcal{S}(D_T)} \qquad \text{(by (2.10)),} \\
&\leq C_{bc}\|r_{bc}[u_{\theta^*}]\|_{\mathcal{B}(\partial D_T)} + C_{int}\|r_{int}[u_{\theta^*}]\|_{\mathcal{S}(D_T)} \qquad \text{(by (3.14)),}
\end{aligned} \tag{3.15}$$

By the fact that $\mathcal{S} \subset L^{p_s}(D_T)$ and $\mathcal{B} \subset L^{p_b}(\partial D_T)$, the definition of the training error (3.10) and the quadrature rule (2.19), we see that,

$$\begin{aligned}
\|r_{int}[u_{\theta^*}]\|^{p_s}_{\mathcal{S}(D_T)} &\approx \sum_{n=1}^{N} w_n |r_{int}[u_{\theta^*}](x_n)|^{p_s} = \left(\mathcal{E}_T^{int}\right)^{p_s}, \\
\|r_{bc}[u_{\theta^*}]\|^{p_b}_{\mathcal{B}(\partial D_T)} &\approx \sum_{m=1}^{M} w_m |r_{bc}[u_{\theta^*}](y_m)|^{p_b} = \left(\mathcal{E}_T^{bc}\right)^{p_b}.
\end{aligned}$$

Hence, the training error is a quadrature for the residual (3.2) and the resulting quadrature error, given by (2.20) translates to,

$$\begin{aligned}
\|r_{int}[u_{\theta^*}]\|^{p_s}_{\mathcal{S}(D_T)} &\leq \left(\mathcal{E}_T^{int}\right)^{p_s} + C^{int}_{quad} N^{-\alpha_s}, \\
\|r_{bc}[u_{\theta^*}]\|^{p_b}_{\mathcal{B}(\partial D_T)} &\leq \left(\mathcal{E}_T^{bc}\right)^{p_b} + C^{bc}_{quad} M^{-\alpha_b}.
\end{aligned} \tag{3.16}$$

Substituting (3.16) into (3.15) yields the desired bound (3.13). □

Observe that the stability estimate (3.12) is a special case of the general estimate (2.10).

The main point of the error estimate (3.13) is to link the total (generalization) error to the training error. As a matter of fact, it is not at all obvious that minimizing the PDE residual (3.2) can lead to any control on the generalization error (3.9). The error estimate (3.13) provides this connection by leveraging the stability of PDEs to derive an error estimate in terms of the PDE residual. Hence, *the theorem rigorously establishes the underlying mechanism by which PINNs can effectively approximate solutions to partial differential equations, and this mechanism is primarily centered on the stability of the forward problem.*

Clearly, the stability of the forward problem is not the unique element to guarantee low generalization error. Indeed, an inspection of the estimate (3.13) reveals that the generalization error for the PINN is small as long as the following hold,

- The training error $\mathcal{E}_T \ll 1$ has to be sufficiently small. Note that we have no a priori control on the training error but can compute it *a posteriori*. We observe that the error estimate (3.13) holds for any function u^*, defined in terms of the residual (3.2), as it only relies on the stability of the underlying PDE and on the accuracy of the underlying quadrature rule. We have not used the structure of neural networks in the proof of Theorem 3.2.1, nor have we used any specific details of the training process. In particular, this estimate (3.13) holds for any neural network of the form (2.24). However, there is no guarantee that the training error $\mathcal{E}_T$ is small for such neural networks. On the other hand, one could expect that the training error for the trained PINN u^* is small.

- The quadrature error depends on the number of quadrature (training) points N, M as well as on the quadrature constants C^{bc}_{quad}, C^{int}_{quad}, which in turn, depends on the residual of the underlying PINN u^* and indirectly, on the number of training points N, M. In particular, it might grow with increasing N and M. Thus, one might need to choose the number of quadrature points N, M large enough such that $C^{int}_{quad}N^{-\alpha_s} \ll 1$, $C^{bc}_{quad}M^{-\alpha_b} \ll 1$. More pertinently, the constants C^{bc}_{quad}, C^{int}_{quad} depends on the architecture of the underlying neural network.

- The constants C_{int}, C_{bc}, that encode the stability of the underlying PDE and depend on both the underlying exact solution u as well as the trained PINN u^* needs to be bounded.

In particular, the evaluation of the bound's constants depends on the details on the underlying PDE and quadrature rule and cannot be worked out in the abstract setup of Theorem 3.2.1. Therefore, in the next sections, we are going to work out details of the stability estimates for a wide class of PDEs, including, semi-linear parabolic PDE, viscous scalar conservation laws, and two examples of dispersive equations.

3.3 Semi-linear Parabolic equations

3.3.1 The underlying PDEs

Let $D \subset \mathbb{R}^d$ be a domain i.e, an open connected bounded set with a C^k boundary ∂D. We consider the following model of a semi-linear parabolic equation,

$$\begin{aligned} u_t &= \Delta u + f(u), \quad \forall x \in D,\ t \in (0,T), \\ u(x,0) &= \bar{u}(x), \quad \forall x \in D, \\ u(x,t) &= 0, \quad \forall x \in \partial D,\ t \in (0,T). \end{aligned} \tag{3.17}$$

Here, $\bar{u} \in H^{\bar{s}}(D;\mathbb{R})$ is the initial data, $u \in H^s(((0,T)\times D);\mathbb{R})$ is the solution, and $f : \mathbb{R} \to \mathbb{R}$ is the non-linear source (reaction) term. We assume that the non-linearity is globally Lipschitz i.e, there exists a constant C_f (independent of v, w) such that

$$|f(v) - f(w)| \leq C_f |v - w|, \quad v, w \in \mathbb{R}. \tag{3.18}$$

In particular, the homogeneous linear heat equation with $f(u) \equiv 0$ and the linear source term $f(u) = c_f u$ are examples of (3.17). Semilinear heat equations with globally Lipschitz non-linearities arise in several models in biology and finance [5].

The existence, uniqueness, and regularity of the semi-linear parabolic equations with Lipschitz non-linearities such as (3.17) can be found in classical textbooks such as [58]. For our purposes here, we will choose $\bar{s} > k + d/2 + 1$ such that the initial data $\bar{u} \in C^k(D)$ and we obtain $u \in C^k([0,T]\times D)$, with $k \geq 2$ as the classical solution of the semi-linear parabolic equation (3.17).

3.3.2 PINNs

In accordance with section 3.1, we will proceed to outline the foundational components of PINNs for a semi-linear parabolic equation.

Training set

Let $D_T = D \times (0, T)$ be the space-time domain. As in section 3.1, we will choose the training set $\mathcal{S}\bar{D}\times \subset [0, T]$, based on suitable quadrature points, corresponding to a suitable space-time grid-based composite Gauss quadrature rule as long $d \leq 3$ or correspond to low-discrepancy sequences for moderately high dimensions or randomly chosen points in very high dimensions.

We then divide the training set into the following three sets,

- Interior training points $\mathcal{S}_{int} = \{y_n\}$ for $1 \leq n \leq N_{int}$, with each $y_n = (x_n, t_n) \in D_T$.
- Spatial boundary training points $\mathcal{S}_{sb} = \{z_n, 0\}$ for $1 \leq n \leq N_{sb}$ with each $z_n = (x_n, t_n)$ and each $x_n \in \partial D$.
- Temporal boundary training points $\mathcal{S}_{tb} = \{x_n, \bar{u}(x_n)\}$, with $1 \leq n \leq N_{tb}$ and each $x_n \in D$.

The full training set is $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$.

Residuals

For the neural network $u_\theta \in C^k(\bar{D} \times [0, T])$, with continuous extensions of the derivatives to the boundaries, defined by (2.24), with a smooth activation function σ and $\theta \in \Theta$ as the set of tuning parameters, we define the following residuals,

- Interior Residual given by,
$$r_{int}[u_\theta](x, t) := \partial_t u_\theta(x, t) - \Delta u_\theta(x, t) - f(u_\theta(x, t)). \tag{3.19}$$
Here $\Delta = \Delta_x$ is the spatial Laplacian. Note that the residual is well defined and $r_{int}[u_\theta] \in C^{k-2}(\bar{D} \times [0, T])$ for every $\theta \in \Theta$.
- Spatial boundary Residual given by,
$$r_{sb}[u_\theta](x, t) := u_\theta(x, t), \quad \forall x \in \partial D,\ t \in (0, T]. \tag{3.20}$$
Given the fact that the neural network is smooth, this residual is well defined.
- Temporal boundary Residual given by,
$$r_{tb}[u_\theta](x) := u_\theta(x, 0) - \bar{u}(x), \quad \forall x \in D. \tag{3.21}$$
Again this quantity is well-defined and $r_{tb}[u_\theta](x) \in C^k(D)$ as both the initial data and the neural network are smooth.

Loss function

We need a loss function to train the PINN. To this end, we set the following loss function,

$$J(\theta) := \sum_{n=1}^{N_{tb}} w_n^{tb} |r_{tb}[u_\theta](x_n)|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb} |r_{sb}[u_\theta](x_n, t_n)|^2 + \sum_{n=1}^{N_{int}} w_n^{int} |r_{int}[u_\theta](x_n, t_n)|^2. \tag{3.22}$$

Here the residuals are defined by (3.21), (3.20), (3.19), w_n^{tb} are the N_{tb} quadrature weights corresponding to the temporal boundary training points $\mathcal{S}_{tb}$, w_n^{sb} are the N_{sb} quadrature weights corresponding to the spatial boundary training points $\mathcal{S}_{sb}$, and w_n^{int} are the N_{int} quadrature weights corresponding to the interior training points $\mathcal{S}_{int}$.

3.3.3 Estimate on the generalization error.

We are interested now in estimating the generalization error (3.9) for a PINN approximating a semi-linear parabolic equation. In this case, $D_T \subset L^2(D \times (0,T))$ and the generalization error is concretely defined as,

$$\mathcal{E}_G := \left(\int_0^T \int_D |u(x,t) - u^*(x,t)|^2 dx dt \right)^{\frac{1}{2}}. \tag{3.23}$$

As for the abstract PDE (2.3), we are going to estimate the generalization error in terms of the *training error* that we define as,

$$\mathcal{E}_T^2 := \underbrace{\sum_{n=1}^{N_{tb}} w_n^{tb} |r_{tb}[u_{\theta^*}](x_n)|^2}_{(\mathcal{E}_T^{tb})^2} + \underbrace{\sum_{n=1}^{N_{sb}} w_n^{sb} |r_{sb}[u_{\theta^*}](x_n,t_n)|^2}_{(\mathcal{E}_T^{sb})^2} + \lambda \underbrace{\sum_{n=1}^{N_{int}} w_n^{int} |r_{int}[u_{\theta^*}](x_n,t_n)|^2}_{(\mathcal{E}_T^{int})^2}. \tag{3.24}$$

We also need the following assumptions on the quadrature error, analogous to (2.10). For any function $g \in C^k(D)$, the quadrature rule corresponding to quadrature weights w_n^{tb} at points $x_n \in \mathcal{S}_{tb}$, with $1 \le n \le N_{tb}$, satisfies

$$\left| \int_D g(x)dx - \sum_{n=1}^{N_{tb}} w_n^{tb} g(x_n) \right| \le C_{quad}^{tb}(\|g\|_{C^k}) N_{tb}^{-\alpha_{tb}}. \tag{3.25}$$

For any function $g \in C^k(\partial D \times [0,T])$, the quadrature rule corresponding to quadrature weights w_n^{sb} at points $(x_n,t_n) \in \mathcal{S}_{sb}$, with $1 \le n \le N_{sb}$, satisfies

$$\left| \int_0^T \int_{\partial D} g(x,t) ds(x) dt - \sum_{n=1}^{N_{sb}} w_n^{sb} g(x_n,t_n) \right| \le C_{quad}^{sb}(\|g\|_{C^k}) N_{sb}^{-\alpha_{sb}}. \tag{3.26}$$

Finally, for any function $g \in C^\ell(D \times [0,T])$, the quadrature rule corresponding to quadrature weights w_n^{int} at points $(x_n,t_n) \in \mathcal{S}_{int}$, with $1 \le n \le N_{int}$, satisfies

$$\left| \int_0^T \int_D g(x,t) dx dt - \sum_{n=1}^{N_{int}} w_n^{int} g(x_n,t_n) \right| \le C_{quad}^{int}(\|g\|_{C^\ell}) N_{int}^{-\alpha_{int}}. \tag{3.27}$$

In the above, $\alpha_{int}, \alpha_{sb}, \alpha_{tb} > 0$ and in principle, different order quadrature rules can be used. We estimate the generalization error for the PINN in the following,

Theorem 3.3.1. *Let $u \in C^k(\bar{D} \times [0,T])$ be the unique classical solution of the semilinear parabolic equation* (3.17) *with the source f satisfying* (3.18)*. Let $u^* = u_{\theta^*}$ be a PINN generated by algorithm 1, corresponding to loss function* (3.22)*. Then the generalization error* (3.23) *can be estimated as,*

$$\mathcal{E}_G \le C_1 \left(\mathcal{E}_T^{tb} + \mathcal{E}_T^{int} + C_2 (\mathcal{E}_T^{sb})^{\frac{1}{2}} + (C_{quad}^{tb})^{\frac{1}{2}} N_{tb}^{-\frac{\alpha_{tb}}{2}} + (C_{quad}^{int})^{\frac{1}{2}} N_{int}^{-\frac{\alpha_{int}}{2}} + C_2 (C_{quad}^{sb})^{\frac{1}{4}} N_{sb}^{-\frac{\alpha_{sb}}{4}} \right), \tag{3.28}$$

with constants given by,

$$\begin{aligned} C_1 &= \sqrt{T + (1+2C_f)T^2 e^{(1+2C_f)T}}, \quad C_2 = \sqrt{2C_{\partial D}(u,u^*)T^{\frac{1}{2}}}, \\ C_{\partial D} &= |\partial D|^{\frac{1}{2}} \left(\|u\|_{C^1([0,T]\times\partial D)} + \|u^*\|_{C^1([0,T]\times\partial D)} \right), \end{aligned} \tag{3.29}$$

and $C^{tb}_{quad} = C^{tb}_{quad}(\|r_{tb}[u_{\theta^*}]^2\|_{C^k})$, $C^{sb}_{quad} = C^{tb}_{quad}(\|r_{sb}[u_{\theta^*}]\|_{C^k})$ *and* $C^{int}_{quad} = C^{int}_{quad}(\|r_{int}[u_{\theta^*}]\|_{C^{k-2}})$ *are the constants defined by the quadrature error* (3.25), (3.26), (3.27), *respectively.*

Proof. By the definitions of the residuals (3.19), (3.20), (3.21) and the underlying PDE (3.17), we can readily verify that the error $\hat{u} := u^* - u$ satisfies the following (forced) parabolic equation,

$$\begin{aligned} \hat{u}_t &= \Delta\hat{u} + f(u^*) - f(u) + r_{int}, \quad \forall x \in D,\ t \in (0,T), \\ \hat{u}(x,0) &= r_{tb}(x), \quad \forall x \in D, \\ u(x,t) &= r_{sb}(x,t), \quad \forall x \in \partial D,\ t \in (0,T). \end{aligned} \tag{3.30}$$

Here, we have denoted $r_{int} = r_{int}[u_{\theta^*}]$ for notational convenience and analogously for the residuals r_{tb}, r_{sb}.

Multiplying both sides of the PDE (3.30) with $\hat{u}$, integrating over the domain and integrating by parts, denoting n as the unit outward normal, yields,

$$\begin{aligned} \frac{1}{2}\frac{d}{dt}\int_D |\hat{u}(x,t)|^2 dx &= -\int_D |\nabla\hat{u}|^2 dx + \int_{\partial D} r_{sb}(x,t)(\nabla\hat{u}\cdot n)ds(x) + \int_D \hat{u}(f(u^*) - f(u))dx + \int_D r_{int}\hat{u}dx. \\ &\le \int_D |\hat{u}||f(u^*) - f(u)|dx + \frac{1}{2}\int_D \hat{u}(x,t)^2 dx + \frac{1}{2}\int_D |r_{int}|^2 dx \\ &+ \underbrace{|\partial D|^{\frac{1}{2}}\left(\|u\|_{C^1([0,T]\times\partial D)} + \|u^*\|_{C^1([0,T]\times\partial D)}\right)}_{C_{\partial D}(u,u^*)}\left(\int_{\partial D} |r_{sb}(x,t)|^2 ds(x)\right)^{\frac{1}{2}} \\ &\overset{(\text{by (3.18)})}{\le} (C_f + \frac{1}{2})\int_D |\hat{u}(x,t)|^2 dx + \frac{1}{2}\int_D |r_{int}|^2 dx + C_{\partial D}(u,u^*)\left(\int_{\partial D} |r_{sb}(x,t)|^2 ds(x)\right)^{\frac{1}{2}}. \end{aligned}$$

Integrating the above inequality over $[0,\bar{T}]$ for any $\bar{T} \le T$ and the definition (3.21) together with Cauchy-Schwarz inequality, we obtain,

$$\begin{aligned} \int_D |\hat{u}(x,\bar{T})|^2 dx \le \int_D |r_{tb}(x)|^2 dx &+ (1 + 2C_f)\int_0^{\bar{T}}\int_D |\hat{u}(x,t)|^2 dxdt + \int_0^T\int_D |r_{int}|^2 dxdt \\ &+ 2C_{\partial D}(u,u^*)T^{\frac{1}{2}}\left(\int_0^T\int_{\partial D} |r_{sb}(x,t)|^2 ds(x)dt\right)^{\frac{1}{2}}. \end{aligned}$$

Applying the integral form of the Grönwall's inequality to the above, we obtain,

$$\begin{aligned} &\int_D |\hat{u}(x,\bar{T})|^2 dx \\ &\le \left(1 + (1 + 2C_f)Te^{(1+2C_f)T}\right)\left(\int_D |r_{tb}(x)|^2 dx + \right. \\ &\left.\int_0^T\int_D |r_{int}|^2 dxdt + C_{\partial D}(u,u^*)T^{\frac{1}{2}}\left(\int_0^T\int_{\partial D} |r_{sb}(x,t)|^2 ds(x)dt\right)^{\frac{1}{2}}\right). \end{aligned}$$

Integrating over $\bar{T} \in [0, T]$ yields,

$$\begin{aligned}
\mathcal{E}_G^2 &= \int_0^T \int_D |\hat{u}(x, \bar{T})|^2 dx dt \\
&\leq \left(T + (1 + 2C_f)T^2 e^{(1+2C_f)T}\right) \left(\int_D |r_{tb}(x)|^2 dx + \right. \\
&\left. + \int_0^T \int_D |r_{int}|^2 dx dt + 2C_{\partial D}(u, u^*) T^{\frac{1}{2}} \left(\int_0^T \int_{\partial D} |r_{sb}(x,t)|^2 ds(x) dt \right)^{\frac{1}{2}} \right).
\end{aligned} \tag{3.31}$$

By the definitions of different components of the training error (3.24) and applying the estimates (3.25), (3.26), (3.27) on the quadrature error yields the desired inequality (3.28). □

Remark 3.3.2. *The estimate* (3.28) *bounds the generalization error in terms of each component of the training error and the quadrature errors. Clearly, each component of the training error can be computed from* (3.24)*, once the training has been completed. As long as each component of the PINN training error is small, the bound* (3.28) *implies that the generalization error will be small for large enough number of training points. Although, the estimate is not by any means sharp as triangle inequalities and the Grönwall's inequality are used, information can be gleaned from it. For instance, the error due to the boundary residual has a bigger weight in* (3.28)*, relative to the interior and initial residuals. This is consistent with the observations of [59] and can also be seen in the recent papers such as [22] and suggests that the loss function* (3.22) *could be modified such that the boundary residual is penalized more.*

Remark 3.3.3. *In addition to the training errors, which could depend on the underlying PDE solution, the estimate* (3.28) *shows explicit dependence on the underlying solution through the constant $C_{\partial D}$, which is based only on the value of the underlying solution on the boundary. Similarly, the dependence on dimension is only seen through the quadrature error.*

3.4 Viscous scalar conservation laws

3.4.1 The underlying PDE

In this section, we consider the following one-dimensional version of *viscous scalar conservation laws* as a model problem for quasilinear, convection-dominated diffusion equations,

$$\begin{aligned}
u_t + f(u)_x &= \nu u_{xx}, \quad \forall x \in (0,1), \ t \in [0, T], \\
u(x, 0) &= \bar{u}(x), \quad \forall x \in (0,1). \\
u(0, t) &= u(1, t) = 0, \quad \forall t \in [0, T].
\end{aligned} \tag{3.32}$$

Here, $\bar{u} \in C^k([0,1])$, for any $k \geq 2$, is the initial data and we consider zero Dirichlet boundary conditions. Note that $0 < \nu \ll 1$ is the viscosity coefficient. The flux function is denoted by $f \in C^k(\mathbb{R}; \mathbb{R})$.

We emphasize that (3.32) is a model problem that we present here for notational and expositional simplicity. The following results can be readily extended in the following directions:

- Several space dimensions.
- Other boundary conditions such as Periodic or Neumann boundary conditions.

- More general forms of the viscous term, namely $\nu\left(B(u)u_x\right)_x$, for any $B \in C^k(\mathbb{R};\mathbb{R})$ with $B(v) \geq c > 0$, for all $v \in \mathbb{R}$ and for some c.

Moreover, we can follow standard textbooks such as [60] to conclude that as long as $\nu > 0$, there exists a classical solution $u \in C^k([0,1] \times [0,T))$ of the viscous scalar conservation law (3.32).

3.4.2 PINNs

We realize the abstract algorithm 1 in the following concrete steps,

Training Set.

Let $D = (0,1)$ and $D_T = (0,1) \times (0,T)$. As in section 3.1, we divide the training set $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$ of the abstract PINNs algorithm 1 into the following three subsets,

- Interior training points $\mathcal{S}_{int} = \{y_n\}$ for $1 \leq n \leq N_{int}$, with each $y_n = (x_n, t_n) \in D_T$. These points can the quadrature points, corresponding to a suitable space-time grid-based composite Gauss quadrature rule or generated from a low-discrepancy sequence in D_T.
- Spatial boundary training points $\mathcal{S}_{sb} = (0, t_n) \cup (1, t_n)$ for $1 \leq n \leq N_{sb}$, and the points t_n chosen either as Gauss quadrature points or low discrepancy sequences in $[0,T]$.
- Temporal boundary training points $\mathcal{S}_{tb} = \{x_n\}$, with $1 \leq n \leq N_{tb}$ and each $x_n \in (0,1)$, chosen either as Gauss quadrature points of low-discrepancy sequences.

Residuals

In algorithm 1 for generating PINNs, we need to define appropriate residuals. For the neural network $u_\theta \in C^k([0,1] \times [0,T])$, defined by (2.24), with a smooth activation function such as $\sigma = \tanh$ and $\theta \in \Theta$ as the set of tuning parameters, we define the following residuals,

- Interior Residual given by,

$$r_{int}[u_\theta](x,t) := \partial_t(u_\theta(x,t)) + \partial_x(f(u_\theta(x,t))) - \nu\partial_{xx}(u_\theta(x,t)). \tag{3.33}$$

- Spatial boundary Residual given by,

$$\begin{aligned} r_{sb,0}[u_\theta](t) &:= u_\theta(0,t), \quad \forall t \in (0,T] \\ r_{sb,1}[u_\theta](t) &:= u_\theta(1,t), \quad \forall t \in (0,T] \end{aligned} \tag{3.34}$$

- Temporal boundary Residual given by,

$$r_{tb}[u_\theta](x) := u_\theta(x,0) - \bar{u}(x), \quad \forall x \in [0,1]. \tag{3.35}$$

All the above quantities are well defined for $k \geq 2$ and $r_{int}[u_\theta] \in C^{k-2}([0,1] \times [0,T]),\ r_{sb}[u_\theta] \in C^k([0,T]),\ r_{tb}[u_\theta] \in C^k([0,1])$.

Loss function

We use the following loss function to train the PINN for approximating the viscous scalar conservation law (3.32),

$$J(\theta) := \sum_{n=1}^{N_{tb}} w_n^{tb}|r_{tb}[u_\theta](x_n)|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb}|r_{sb,0}[u_\theta](t_n)|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb}|r_{sb,1}[u_\theta](t_n)|^2 + \lambda \sum_{n=1}^{N_{int}} w_n^{int}|r_{int}[u_\theta](x_n,t_n)|^2. \tag{3.36}$$

Here the residuals are defined by (3.35), (3.34), (3.33). w_n^{tb} are the N_{tb} quadrature weights corresponding to the temporal boundary training points $\mathcal{S}_{tb}$, w_n^{sb} are the N_{sb} quadrature weights corresponding to the spatial boundary training points $\mathcal{S}_{sb}$ and w_n^{int} are the N_{int} quadrature weights corresponding to the interior training points $\mathcal{S}_{int}$. Furthermore, λ is a hyperparameter for balancing the residuals, on account of the PDE and the initial and boundary data, respectively.

3.4.3 Estimate on the generalization error.

As for the semilinear parabolic equation, we will try to estimate the following generalization error for the PINN $u^* = u_{\theta^*}$, generated through algorithm 1, with loss function (3.36), for approximating the solution of the viscous scalar conservation law (3.32):

$$\mathcal{E}_G := \left(\int_0^T \int_0^1 |u(x,t) - u^*(x,t)|^2 dxdt \right)^{\frac{1}{2}}. \tag{3.37}$$

This generalization error will be estimated in terms of the *training error*,

$$\begin{aligned} \mathcal{E}_T^2 := & \underbrace{\sum_{n=1}^{N_{int}} w_n^{int}|r_{int}[u_{\theta^*}](x_n,t_n)|^2}_{(\mathcal{E}_T^{int})^2} + \underbrace{\sum_{n=1}^{N_{tb}} + w_n^{tb}|r_{tb}[u_{\theta^*}](x_n)|^2}_{(\mathcal{E}_T^{tb})^2} \\ & + \underbrace{\sum_{n=1}^{N_{sb}} w_n^{sb}|r_{sb,0}[u_{\theta^*}](t_n)|^2}_{(\mathcal{E}_T^{sb,0})^2} + \underbrace{\sum_{n=1}^{N_{sb}} w_n^{sb}|r_{sb,1}[u_{\theta^*}](t_n)|^2}_{(\mathcal{E}_T^{sb,1})^2}, \end{aligned} \tag{3.38}$$

readily computed from the training loss (3.36) *a posteriori*. We have the following estimate,

Theorem 3.4.1. *Let $\nu > 0$ and let $u \in C^k((0,1) \times (0,T))$ be the unique classical solution of the viscous scalar conservation law* (3.32)*. Let $u^* = u_{\theta^*}$ be the PINN, generated by algorithm 1, with loss function* (3.36)*. Then, the generalization error* (3.37) *is bounded by,*

$$\begin{aligned} \mathcal{E}_G^2 \leq & \left(T + CT^2 e^{CT}\right) \left[\left(\mathcal{E}_T^{tb}\right)^2 + \left(\mathcal{E}_T^{int}\right)^2 + 2\bar{C}_b \left(\left(\mathcal{E}_T^{sb,0}\right)^2 + \left(\mathcal{E}_T^{sb,1}\right)^2 \right) + 2\nu C_b T^{\frac{1}{2}} \left(\mathcal{E}_T^{sb,0} + \mathcal{E}_T^{sb,1} \right) \right] \\ & + \left(T + CT^2 e^T\right) \left[C_{quad}^{tb} N_{tb}^{-\alpha_{tb}} + C_{quad}^{int} N_{int}^{-\alpha_{int}} + 2\bar{C}_b \left(\left(C_{quad}^{sb,0} + C_{quad}^{sb,1} \right) N_{sb}^{-\alpha_{sb}} \right) \right. \\ & \left. + 2\nu C_b T^{\frac{1}{2}} \left(\left(C_{quad}^{sb,0} + C_{quad}^{sb,1} \right)^{\frac{1}{2}} N_{sb}^{-\frac{\alpha_{sb}}{2}} \right) \right]. \end{aligned} \tag{3.39}$$

Here, the training errors are defined by (3.38) *and the constants are given by* $C = 1 + 2C_{f,u,u^*}$, *with*

$$\begin{aligned} C_{f,u,u^*} &= C\left(\|f\|_{C^2}, \|u\|_{W^{1,\infty}}, \|u^*\|_{L^\infty}\right) = |f''\left(\max\{\|u\|_{L^\infty}, \|u^*\|_{L^\infty}\}\right)| \, \|u_x\|_{L^\infty}, \\ C_b &= \left(\|u_x\|_{C([0,1]\times[0,T])} + \|u_x^*\|_{C([0,1]\times[0,T])}\right), \end{aligned} \tag{3.40}$$

$\bar{C}_b = \bar{C}_b\left(\|f'\|_\infty, \|u^*\|_{C^0([0,1]\times[0,T])}\right)$ *and* $C_{quad}^{tb} = C_{quad}^{tb}\left(\|r_{tb}[u_{\theta^*}]\|_{C^k}\right)$, $C_{quad}^{int} = C_{quad}^{int}\left(\|r_{int}[u_{\theta^*}]\|_{C^{k-2}}\right)$, $C_{quad}^{sb,0} = C_{quad}^{sb,0}\left(\|r_{sb,0}[u_{\theta^*}]^2\|_{C^k}\right)$, $C_{quad}^{sb,1} = C_{quad}^{sb,1}\left(\|r_{sb,0}[u_{\theta^*}]^2\|_{C^k}\right)$ *are the constants that appear in the bounds on quadrature error* (3.25)-(3.27).

Proof. We drop the θ^*-dependence of the residuals (3.33)-(3.35) for notational convenience in the following. Define the *entropy flux function*,

$$Q(u) = \int_a^u s f'(s) ds, \tag{3.41}$$

for any $a \in \mathbb{R}$. Let $\hat{u} = u^* - u$ be the error with the PINN. From the PDE (3.32) and the definition of the interior residual (3.33), we have the following identities,

$$\begin{aligned} \partial_t\left(\frac{(u^*)^2}{2}\right) + \partial_x Q(u^*) &= \nu u^* u_{xx}^* + r_{int} u^* \\ \partial_t\left(\frac{u^2}{2}\right) + \partial_x Q(u) &= \nu u u_{xx} \end{aligned} \tag{3.42}$$

A straightforward calculation with (3.32) and (3.33) yields,

$$\partial_t(u\hat{u}) + \partial_x\left(u\left(f(u^*) - f(u)\right)\right) = \left[f(u^*) - f(u) - f'(u)\hat{u}\right] u_x + r_{int} u + \nu\left(u\hat{u}_{xx} + \hat{u}u_{xx}\right). \tag{3.43}$$

Subtracting the second equation of (3.42) and (3.43) from the first equation of (3.42) yields,

$$\partial_t S(u, u^*) + \partial_x H(u, u^*) = r_{int}\hat{u} + T_1 + T_2, \tag{3.44}$$

with,

$$\begin{aligned} S(u, u^*) &:= \frac{(u^*)^2}{2} - \frac{u^2}{2} - \hat{u}u = \frac{1}{2}\hat{u}^2, \\ H(u, u^*) &:= Q(u^*) - Q(u) - u(f(u^*) - f(u)), \\ T_1 &= -\left[f(u^*) - f(u) - f'(u)\hat{u}\right] u_x, \\ T_2 &= \nu\left(u^* u_{xx}^* - u u_{xx} - u\hat{u}_{xx} - \hat{u}u_{xx}\right) = \nu\hat{u}\hat{u}_{xx}. \end{aligned}$$

As the flux f is smooth, by a Taylor expansion, we see that

$$T_1 = -f''(u + \gamma(u^* - u))\hat{u}^2 u_x, \tag{3.45}$$

for some $\gamma \in (0, 1)$. Hence, a straightforward estimate for T_1 is given by,

$$|T_1| \le C_{f,u,u^*}\hat{u}^2, \tag{3.46}$$

with C_{f,u,u^*} defined in (3.40). Next, we integrate (3.44) over the domain $(0, 1)$ and integrate by parts to obtain,

$$\begin{aligned} \frac{d}{dt}\int_0^1 \hat{u}^2(x,t)dx \le{}& 2H(u(0,t), u^*(0,t)) - 2H(u(1,t), u^*(1,t)) \\ &+ C\int_0^1 \hat{u}^2(x,t)dx + \int_0^1 r_{int}^2(x,t)dx, \\ &- 2\nu\int_0^1 \hat{u}_x^2(x,t)dx + 2\nu\left(\hat{u}(1,t)\hat{u}_x(1,t) - \hat{u}(0,t)\hat{u}_x(0,t)\right), \end{aligned} \tag{3.47}$$

with the constant, $C = 1 + 2C_{f,u,u^*}$.

Next, for any $\bar{T} \le T$, we estimate the boundary terms starting with,

$$\begin{aligned}
\int_0^{\bar{T}} \hat{u}(0,t)\hat{u}_x(0,t)dt &= \int_0^{\bar{T}} r_{sb,0}(t)\left(u_x^*(0,t) - u_x(0,t)\right)dt \\
&\le \underbrace{\left(\|u_x\|_{C([0,1]\times[0,T])} + \|u_x^*\|_{C([0,1]\times[0,T])}\right)}_{C_b} T^{\frac{1}{2}} \left(\int_0^T r_{sb,0}^2(t)dt\right)^{\frac{1}{2}}.
\end{aligned}$$

Analogously we can estimate,

$$\int_0^{\bar{T}} \hat{u}(1,t)\hat{u}_x(1,t)dt \le C_b T^{\frac{1}{2}} \left(\int_0^T r_{sb,1}^2(t)dt\right)^{\frac{1}{2}}.$$

We can also estimate from (3.32) and (3.20) that,

$$\begin{aligned}
H(u(0,t),u^*(0,t)) &= Q(u^*(0,t)) - Q(u(0,t)) - u(0,t)(f(u^*(0,t)) - f(u(0,t))), \\
&= Q(r_{sb,0}(t)) - Q(0), \quad \text{as } u(0,t) = 0, \\
&= Q'(\gamma_0 r_{sb,0}(t))r_{sb,0}(t), \quad \text{for some } \gamma_0 \in (0,1), \\
&= \gamma_0 f'(\gamma_0 u^*(0,t))r_{sb,0}^2(t), \quad \text{by (3.41)}, \\
&\le \bar{C}_b r_{sb,0}^2(t), \quad \text{with } \bar{C}_b = \bar{C}_b\left(\|f'\|_\infty, \|u^*\|_{C^0([0,1]\times[0,T])}\right).
\end{aligned}$$

Analogously, we can estimate,

$$H(u(1,t),u^*(1,t)) \le \bar{C}_b r_{sb,1}^2(t).$$

For any $\bar{T} \le T$, integrating (3.47) over the time interval $[0,\bar{T}]$ and using the above inequalities on the boundary terms, together with the definition of the residual (3.35) yields,

$$\begin{aligned}
&\int_0^1 \hat{u}^2(x,\bar{T})dx \le \mathcal{C} + C\int_0^{\bar{T}}\int_0^1 \hat{u}^2(x,t)dxdt, \\
&\quad \mathcal{C} = \int_0^1 r_{tb}^2(x)dx + \int_0^T\int_0^1 r_{int}^2(x,t)dxdt \\
&\quad + 2\bar{C}_b\left[\int_0^T r_{sb,0}^2(t)dt + \int_0^T r_{sb,1}^2(t)dt\right] + 2\nu C_b T^{\frac{1}{2}}\left[\left(\int_0^T r_{sb,0}^2(t)dt\right)^{\frac{1}{2}} + \left(\int_0^T r_{sb,1}^2(t)dt\right)^{\frac{1}{2}}\right].
\end{aligned} \tag{3.48}$$

By applying the integral form of the Grönwall's inequality to (3.48) for any $\bar{T} \le T$ and integrating again over $\bar{T}$, together with the definition of the generalization error (3.37), we obtain,

$$\mathcal{E}_G^2 \le \left(T + CT^2 e^{CT}\right)\mathcal{C}. \tag{3.49}$$

Using the bounds (3.25)-(3.27) on the quadrature errors and the definition of $\mathcal{C}$ in (3.48), we obtain,

$$\mathcal{C} \le \sum_{n=1}^{N_{tb}} w_n^{tb}|r_{tb}(x_n)|^2 + C_{quad}^{tb}\left(\|r_{tb}\|_{C^k}\right)N_{tb}^{-\alpha_{tb}}$$

$$
\begin{aligned}
&+ \sum_{n=1}^{N_{int}} w_n^{int} |r_{int}(x_n,t_n)|^2 + C_{quad}^{int}\left(\|r_{int}\|_{C^{k-2}}\right) N_{int}^{-\alpha_{int}}, \\
&+ 2\bar{C}_b \left[\sum_{n=1}^{N_{sb}} w_n^{sb} |r_{sb,0}(t_n)|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb} |r_{sb,1}(t_n)|^2 + \left(C_{quad}^{sb}\left(\|r_{sb,0}\|_{C^k}\right) + C_{quad}^{sb}\left(\|r_{sb,1}\|_{C^k}\right)\right) N_{sb}^{-\alpha_{sb}} \right] \\
&+ 2\nu C_b T^{\frac{1}{2}} \left[\left(\sum_{n=1}^{N_{sb}} w_n^{sb} |r_{sb,0}(t_n)|^2 \right)^{\frac{1}{2}} + \left(\sum_{n=1}^{N_{sb}} w_n^{sb} |r_{sb,1}(t_n)|^2 \right)^{\frac{1}{2}} + \right. \\
&\left. + \left(C_{quad}^{sb}\left(\|r_{sb,0}\|_{C^k}\right) + C_{quad}^{sb}\left(\|r_{sb,1}\|_{C^k}\right)\right)^{\frac{1}{2}} N_{sb}^{-\frac{\alpha_{sb}}{2}} \right].
\end{aligned}
$$

From definition of training errors (3.38) and (3.49) and the above inequality, we obtain the desired estimate (3.39). ☐

Remark 3.4.2. *The estimate* (3.39) *is a concrete realization of the abstract estimate* (3.13)*, with training error decomposed into 4 parts, the constants, associated with the PDE, are given by* C_{f,u,u^*}, C_b (3.40) *and the constants due to the quadrature errors are also clearly delineated.*

Remark 3.4.3. *A close inspection of the estimate* (3.39) *reveals that at the very least, the classical solution u of the PDE* (3.32) *needs to be in* $L^\infty((0,T);W^{1,\infty}((0,1)))$ *for the rhs of* (3.39) *to be bounded. This indeed holds as long as* $\nu > 0$. *However, it is well known (see [60] and references therein) that if* u^ν *is the solution of* (3.32) *for viscosity* ν, *then for some initial data,*

$$
\|u^\nu\|_{L^\infty((0,T);W^{1,\infty}((0,1)))} \sim \frac{1}{\sqrt{\nu}}. \tag{3.50}
$$

Thus, in the limit $\nu \to 0$, *the constant* C_{f,u,u^*} *can blow up (exponentially in time) and the bound* (3.39) *no longer controls the generalization error. This is not unexpected as the whole strategy of this paper relies on pointwise realization of residuals. However, the zero-viscosity limit of* (3.32)*, leads to a scalar conservation law with discontinuous solutions (shocks) and the residuals are measures that do not make sense pointwise. Thus, the estimate* (3.39) *also points out the limitations of a PINN for approximating discontinuous solutions.*

An alternative strategy has to be pursued in order approximate solutions to inviscid scalar conservation laws ($\nu = 0$). This will be formally addressed in Chapter 4.

3.5 Incompressible Euler Equations

3.5.1 The underlying PDE

The motion of an inviscid, incompressible fluid is modeled by the incompressible Euler equations [61]. We consider the following form of these PDEs,

$$
\begin{aligned}
u_t + (u\cdot\nabla)\, u + \nabla p = f, &\quad (x,t) \in D \times (0,T), \\
\nabla \cdot u = 0, &\quad (x,t) \in D \times (0,T), \\
u \cdot n = 0, &\quad (x,t) \in \partial D \times (0,T), \\
u(x,0) = \bar{u}(x), &\quad x \in D.
\end{aligned} \tag{3.51}
$$

Here, $D \subset \mathbb{R}^d$, for $d = 2, 3$ is an open, bounded, connected subset with smooth C^1 boundary ∂D, $D_T = D \times (0, T)$, $u : D_T \to \mathbb{R}^d$ is the velocity field, $p : D_T \to \mathbb{R}$ is the pressure that acts as a Lagrange multiplier to enforce the divergence constraint and $f \in C^1(D_T; \mathbb{R}^d)$ is a forcing term. We use the *no penetration* boundary conditions here with n denoting the unit outward normal to ∂D.

Note that we have chosen to present this form of the incompressible Euler equations for simplicity of exposition. The analysis, presented below, can be readily but tediously extended to the following,

- Other boundary conditions such as periodic boundary conditions on the torus $\mathbb{T}^d$.
- The *Navier-Stokes equations*, where we add the *viscous term* $\nu\Delta u$ to the first equation in (3.51), with either periodic boundary conditions or the so-called *no slip* boundary conditions i.e, $u \equiv 0$, for all $x \in \partial D$ and for all $t \in (0, T]$.

3.5.2 PINNs

We describe the algorithm 1 for this PDE in the following steps,

Training set

We chose the training set $\mathcal{S} \subset D_T$ with $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$, with interior, spatial and temporal boundary training sets, chosen exactly as in section 3.3.2, either as quadrature points for a (composite) Gauss rule or as low-discrepancy sequences on the underlying domains.

Residuals

For the neural networks $(x, t) \mapsto (u_\theta(x, t), p_\theta(x, t)) \in C^k(D \times (0, T)) \cap C(\bar{D} \times [0, T])$, defined by (2.24), with a smooth activation function and $\theta \in \Theta$ as the set of tuning parameters, we define the residual r in algorithm 1, consisting of the following parts,

- *Velocity residual* given by,

$$r_u[u_\theta](x, t) := (u_\theta)_t + (u_\theta \cdot \nabla)\, u_\theta + \nabla p_\theta - f, \quad (x, t) \in D \times (0, T), \tag{3.52}$$

- *Divergence residual* given by,

$$r_{div}[u_\theta](x, t) := \nabla \cdot u_\theta(x, t), \quad (x, t) \in D \times (0, T), \tag{3.53}$$

- *Spatial boundary Residual* given by,

$$r_{sb}[u_\theta](x, t) := u_\theta(x, t) \cdot n, \quad \forall x \in \partial D, \ t \in (0, T]. \tag{3.54}$$

- *Temporal boundary Residual* given by,

$$r_{tb}[u_\theta](x) := u_\theta(x, 0) - \bar{u}(x), \quad \forall x \in D. \tag{3.55}$$

As the underlying neural networks have the required regularity, the residuals are well-defined.

Loss function

We consider the following loss function for training PINNs to approximate the incompressible Euler equation (3.51),

$$J(\theta) := \sum_{n=1}^{N_{tb}} w_n^{tb}|r_{tb}[u_\theta](x_n)|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb}|r_{sb}[u_\theta](x_n,t_n)|^2 + \lambda\left(\sum_{n=1}^{N_{int}} w_n^{int}|r_u[u_\theta](x_n,t_n)|^2 + \sum_{n=1}^{N_{int}} w_n^{int}|r_{div}[u_\theta](x_n,t_n)|^2\right). \tag{3.56}$$

Here the residuals are defined by (3.52)-(3.55). w_n^{tb} are the N_{tb} quadrature weights corresponding to the temporal boundary training points $\mathcal{S}_{tb}$, w_n^{sb} are the N_{sb} quadrature weights corresponding to the spatial boundary training points $\mathcal{S}_{sb}$ and w_n^{int} are the N_{int} quadrature weights corresponding to the interior training points $\mathcal{S}_{int}$. Furthermore, λ is a hyperparameter for balancing the residuals, on account of the PDE and the initial and boundary data, respectively.

3.5.3 Estimate on the generalization error.

We denote the PINN, obtained by the algorithm 1, for approximating the incompressible Euler equations, as $u^* = u_{\theta^*}$, with θ^* being a (approximate local) minimum of the loss function (3.56). We consider the following generalization error,

$$\mathcal{E}_G := \left(\int_0^T \int_D \|u(x,t) - u^*(x,t)\|^2 dx dt\right)^{\frac{1}{2}}, \tag{3.57}$$

with $\|\cdot\|$ denoting the Euclidean norm in $\mathbb{R}^d$. Note that we only consider the error with respect to the velocity field u in (3.57). Although the pressure p in (3.51) is approximated by the neural network $p^* = p_{\theta^*}$, we recall that the pressure is a Lagrange multiplier, and not a primary variable in the incompressible Euler equations. Hence, we will not consider pressure errors here.

As in section 3.2, we will bound the generalization error in terms of the following *training errors*,

$$\mathcal{E}_T^2 := \underbrace{\sum_{n=1}^{N_{tb}} w_n^{tb}|r_{tb}[u_{\theta^*}](x_n)|^2}_{(\mathcal{E}_T^{tb})^2} + \underbrace{\sum_{n=1}^{N_{sb}} w_n^{sb}|r_{sb}[u_{\theta^*}](x_n,t_n)|^2}_{(\mathcal{E}_T^{sb})^2} + \underbrace{\sum_{n=1}^{N_{int}} w_n^{int}|r_u[u_{\theta^*}](x_n,t_n)|^2}_{(\mathcal{E}_T^{u})^2} + \lambda \underbrace{\sum_{n=1}^{N_{int}} w_n^{int}|r_{div}[u_{\theta^*}](x_n,t_n)|^2}_{(\mathcal{E}_T^{d})^2}. \tag{3.58}$$

As in the previous sections, the training errors can be readily computed *a posteriori* from the loss function (3.6), (3.56).

We have the following bound on the generalization error in terms of the training error,

Theorem 3.5.1. *Let $u \in C^1(D \times (0,T)) \cap C(\bar{D} \times [0,T])$ be the classical solution of the incompressible Euler equations (3.51). Let $u^* = u_{\theta^*}, p^* = p_{\theta^*}$ be the PINN generated by algorithm 1, then the resulting generalization error (3.57) is bounded as,*

$$\begin{aligned}\mathcal{E}_G^2 &\leq \left(T + C_\infty T^2 e^{C_\infty T}\right)\left[\left(\mathcal{E}_T^{tb}\right)^2 + \left(\mathcal{E}_T^{u}\right)^2 + C_0 T^{\frac{1}{2}}\left(\mathcal{E}_T^{div} + \mathcal{E}_T^{sb}\right)\right] \\ &+ \left(T + C_\infty T^2 e^{C_\infty T}\right)\left[C_{quad}^{tb} N_{tb}^{-\alpha_{tb}} + C_{quad}^{u} N_{int}^{-\alpha_{int}} + \left(C_{quad}^{div}\right)^{\frac{1}{2}} N_{int}^{-\frac{\alpha_{int}}{2}} + \left(C_{quad}^{sb}\right)^{\frac{1}{2}} N_{sb}^{-\frac{\alpha_{sb}}{2}}\right].\end{aligned} \tag{3.59}$$

Here, the training errors are defined in (3.58) *and the constants are given by,*

$$\begin{aligned} C_0 &= C\left(\|u\|_{C^0([0,T]\times\bar{D})}, \|u^*\|_{C^0([0,T]\times\bar{D})}, \|p\|_{C^0([0,T]\times\bar{D})}, \|p^*\|_{C^0([0,T]\times\bar{D})}\right), \\ C_\infty &= 1 + 2C_d\|\nabla u\|_{L^\infty(D_T)}, \end{aligned} \tag{3.60}$$

with C_d only depending on dimension d and $C^{tb}_{quad} = C^{tb}_{quad}\left(\|r_{tb}[u_{\theta^*}]^2\|_{C^k}\right)$, $C^{u}_{quad} = C^{int}_{quad}\left(\|r_u[u_{\theta^*}]^2\|_{C^{k-1}}\right)$, $C^{div}_{quad} = C^{int}_{quad}\left(\|r_{div}[u_{\theta^*}]^2\|_{C^{k-1}}\right)$ and $C^{sb}_{quad} = C^{sb}_{quad}\left(\|r_{sb}[u_{\theta^*}]^s\|_{C^k}\right)$ are the constants associated with the quadrature errors (3.25)-(3.27)*.*

Proof. We will drop explicit dependence of all quantities on the parameters θ^* for notational convenience. We denote the difference between the underlying solution u of (3.51) and PINN u^* as $\hat{u} = u^* - u$. Similarly $\hat{p} = p^* - p$. Using the PDE (3.51) and the definitions of the residuals (3.52)-(3.55), a straightforward calculation yields the following PDE for the $\hat{u}$,

$$\begin{aligned} \hat{u}_t + (\hat{u}\cdot\nabla)\,\hat{u} + (u\cdot\nabla)\,\hat{u} + (\hat{u}\cdot\nabla)\,u + \nabla\hat{p} &= r_u, \quad (x,t)\in D\times(0,T), \\ \nabla\cdot\hat{u} &= r_{div}, \quad (x,t)\in D\times(0,T), \\ \hat{u}\cdot n &= r_{sb}, \quad (x,t)\in \partial D\times(0,T), \\ u(x,0) &= r_{tb}, \quad x\in D. \end{aligned} \tag{3.61}$$

We take a inner product of the first equation in (3.61) with the vector $\hat{u}$ and use the following vector identities,

$$\begin{aligned} \hat{u}\cdot\partial_t\hat{u} &= \partial_t\left(\frac{\|\hat{u}\|^2}{2}\right), \\ \hat{u}\cdot((\hat{u}\cdot\nabla)\,\hat{u}) &= (\hat{u}\cdot\nabla)\left(\frac{\|\hat{u}\|^2}{2}\right), \\ \hat{u}\cdot((u\cdot\nabla)\,\hat{u}) &= (u\cdot\nabla)\left(\frac{\|\hat{u}\|^2}{2}\right), \end{aligned}$$

yields the following identity,

$$\partial_t\left(\frac{\|\hat{u}\|^2}{2}\right) + (\hat{u}\cdot\nabla)\left(\frac{\|\hat{u}\|^2}{2}\right) + (u\cdot\nabla)\left(\frac{\|\hat{u}\|^2}{2}\right) + \hat{u}\cdot((\hat{u}\cdot\nabla)\,u) + (\hat{u}\cdot\nabla)\,\hat{p} = \hat{u}\cdot r_u.$$

Integrating the above identity over D and integrating by parts, together with (3.51) and (3.61) yields,

$$\begin{aligned} \frac{d}{dt}\int_D\left(\frac{\|\hat{u}\|^2}{2}\right)dx &= \int_D r_{div}\left(\frac{\|\hat{u}\|^2}{2} + \hat{p}\right)dx - \int_{\partial D} r_{sb}\left(\frac{\|\hat{u}\|^2}{2} + \hat{p}\right)ds(x) \\ &\quad - \int_D \hat{u}\cdot((\hat{u}\cdot\nabla)\,u)\,dx + \int_D \hat{u}\cdot r_u dx. \end{aligned} \tag{3.62}$$

It is straightforward to obtain the following inequality,

$$\int_D \hat{u}\cdot((\hat{u}\cdot\nabla)\,u)\,dx \le C_d\|\nabla u\|_\infty \int_D \|\hat{u}\|^2 dx,$$

with the constant C_d only depending on dimension and $\|u\|_\infty = \|u\|_{L^\infty(D_T)}$.

Using the above estimate and estimating (3.62) yields,

$$\frac{d}{dt}\int_D \|\hat{u}\|^2 dx \le C_0\left[\left(\int_D (r_{div})^2\,dx\right)^{\frac{1}{2}} + \left(\int_{\partial D}(r_{sb})^2\,ds(x)\right)^{\frac{1}{2}}\right] + C_\infty\int_D\|\hat{u}\|^2 dx + \int_D r_u^2 dx, \tag{3.63}$$

with constants given by (3.60).

For any $\bar{T} \leq T$, we integrate (3.63) over time and use some simple inequalities to obtain,

$$\begin{aligned}\int_D \|\hat{u}(x,\bar{T})\|^2 dx &\leq \mathcal{C} + C_\infty \int_0^{\bar{T}} \int_D \|\hat{u}(x,t)\|^2 dxdt,\\ \mathcal{C} &= \int_D r_{tb}^2 dx + \int_0^T \int_D r_u^2 dxdt,\\ &+ C_0 T^{\frac{1}{2}} \left[\left(\int_0^T \int_D (r_{div})^2 \, dxdt \right)^{\frac{1}{2}} + \left(\int_0^T \int_{\partial D} (r_{sb})^2 \, ds(x)dt \right)^{\frac{1}{2}} \right].\end{aligned} \tag{3.64}$$

Now by using the integral form of the Grönwall's inequality in (3.64) and integrating again over $[0,T]$ results in,

$$\mathcal{E}_G^2 \leq \left(T + C_\infty T^2 e^{C_\infty T}\right) \mathcal{C}. \tag{3.65}$$

Using the bounds (3.25)-(3.27) on the quadrature errors and the definition of $\mathcal{C}$ in (3.64), we obtain,

$$\begin{aligned}\mathcal{C} &\leq \sum_{n=1}^{N_{tb}} w_n^{tb} |r_{tb}(x_n)|^2 + C_{quad}^{tb} \left(\|r_{tb}\|_{C^k}\right) N_{tb}^{-\alpha_{tb}}\\ &+ \sum_{n=1}^{N_{int}} w_n^{int} |r_u(x_n,t_n)|^2 + C_{quad}^{int} \left(\|r_u\|_{C^{k-1}}\right) N_{int}^{-\alpha_{int}},\\ &+ C_0 T^{\frac{1}{2}} \left[\left(\sum_{n=1}^{N_{int}} w_n^{int} |r_{div}(x_n,t_n)|^2 \right)^{\frac{1}{2}} + \left(C_{quad}^{int} \left(\|r_{div}\|_{C^{k-1}}\right)\right)^{\frac{1}{2}} N_{int}^{-\frac{\alpha_{int}}{2}} \right]\\ &+ C_0 T^{\frac{1}{2}} \left[\left(\sum_{n=1}^{N_{sb}} w_n^{sb} |r_{sb}(x_n,t_n)|^2 \right)^{\frac{1}{2}} + \left(C_{quad}^{sb} \left(\|r_{sb}\|_{C^k}\right)\right)^{\frac{1}{2}} N_{sb}^{-\frac{\alpha_{sb}}{2}} \right]\end{aligned}$$

From definition of training errors (3.58) and (3.65) and the above inequality, we obtain the desired estimate (3.59). □

The bound (3.59) explicitly requires the existence of a classical solution u to the incompressible Euler equations, with a minimum regularity of $\nabla u \in L^\infty(D \times (0,T))$. Such solutions do exist as long as we consider the incompressible Euler equations in *two space dimensions* and with sufficiently smooth initial data [61]. However, in three space dimensions, even with smooth initial data, the existence of smooth solutions is a major open question. It is possible that the derivative blows up and the constant C_∞ in (3.59) is unbounded, leading to a loss of control on the generalization error. In general, complicated solutions of the Euler equations are characterized by strong vorticity, resulting in large values of the spatial derivative. The bound (3.59) makes it clear that the generalization error with PINNs can be large for such problems.

3.6 Korteweg de-Vries and Kawahara equations

We will finally apply the PINNs algorithm 1 to the well-known KdV-Kawahara equations, which is an example of dispersive equation.

3.6.1 The underlying PDEs

The general form of the KdV-Kawahara equation is given by,

$$\begin{aligned}
u_t + uu_x + \alpha u_{xxx} - \beta u_{xxxxx} &= 0, \quad \forall\, x \in (0,1),\, t \in (0,T),\\
u(x,0) &= \bar{u}(x), \quad \forall\, x \in (0,1),\\
u(0,t) &= h_1(t), \quad \forall\, t \in (0,T),\\
u(1,t) &= h_2(t), \quad \forall\, t \in (0,T),\\
u_x(0,t) &= h_3(t), \quad \forall\, t \in (0,T),\\
u_x(1,t) &= h_4(t), \quad \forall\, t \in (0,T),\\
u_{xx}(1,t) &= h_5(t), \quad \forall\, t \in (0,T).
\end{aligned} \tag{3.66}$$

Here α, β are non-negative real constants. Note that if $\beta = 0$, then the above equation is called Korteweg de-Vries (KdV) equation, and if $\beta \neq 0$, then the above equation is called the Kawahara equation. It is well known that KdV equation plays a pivotal role in the modeling of shallow water waves, and in particular, the one-dimensional waves of small but finite amplitude in dispersive systems. However, under certain circumstances, the coefficient of the third order derivative in the KdV equation may become very small or even zero [62]. In such a scenario, one has to take account of the higher order effect of dispersion in order to balance the nonlinear effect, which leads to the Kawahara equation.

For the sake of simplicity it will be assumed $\alpha = \beta = 1$ in the upcoming analysis, since their values are not relevant in the present setting, while emphasizing that that the subsequent analysis also holds for the case $\beta = 0$ (KdV equations). Regarding the existence and stability of solutions to (3.66), we closely follow the work by Faminskii & Larkin [63], and recall the following result.

Theorem 3.6.1. *For any integer $k \geq 0$, $n \in \mathbb{N}$, $l = 1$ or 2, define the spaces*

$$\begin{aligned}
\mathcal{X}_k((0,1)\times(0,T)) &:= \Big\{ u : \partial_t^n u \in C([0,T]; H^{5(k-n)}(0,1)) \cap L^2((0,T); H^{5(k-n)+1}(0,1)) \Big\},\\
\mathcal{B}_k^l(0,T) &:= \prod_{j=0}^{l} H^{k+(2-j)/5}(0,T).
\end{aligned}$$

Let $\bar{u} \in H^{5k}(0,1)$, boundary data $(h_1,h_3) \in \mathcal{B}_k^1(0,T)$, and $(h_2,h_4,h_5) \in \mathcal{B}_k^2(0,T)$ satisfy the natural compatibility conditions. Then there exists a unique solution $u \in \mathcal{X}_k$, and the flow map is Lipschitz continuous on any ball in the corresponding norm.

By choosing appropriate values of k (for our purpose, we take $k = 2$) in the above theorem, we readily infer the existence of classical solutions of the Kawahara equations (3.66) by the embedding of Sobolev spaces in the C^ℓ spaces.

3.6.2 PINNs

We apply algorithm 1 to approximate the solutions of (3.66). To this end, we need the following steps,

Training Set.

Let us define the space-time domain $D_T = (0,1) \times (0,T)$, and divide the training set $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$ of the abstract PINNs algorithm 1 into the following three subsets,

- Interior training points $\mathcal{S}_{int} = \{y_n\}$ for $1 \le n \le N_{int}$, with each $y_n = (x_n, t_n) \in D_T$. We use low-discrepancy Sobol points as training points.
- Spatial boundary training points $\mathcal{S}_{sb} = (0, t_n) \cup (1, t_n)$ for $1 \le n \le N_{sb}$, and the points t_n chosen as low-discrepancy Sobol points.
- Temporal boundary training points $\mathcal{S}_{tb} = \{x_n\}$, with $1 \le n \le N_{tb}$ and each $x_n \in (0,1)$, chosen as low-discrepancy Sobol points.

Residuals

To define residuals for the neural network $u_\theta \in C^5([0,T] \times [0,1])$, defined by (2.24), with $\theta \in \Theta$ as the set of tuning parameters, we use the hyperbolic tangent tanh activation function, i.e., $\sigma = \tanh$. With this setting, we define the following residuals

- Interior Residual given by,

$$r_{int}[u_\theta](x,t) := \partial_t u_\theta(x,t) + u_\theta (u_\theta)_x(x,t) + (u_\theta)_{xxx}(x,t) - (u_\theta)_{xxxxx}(x,t). \tag{3.67}$$

  Note that the above residual is well-defined and $r_{int}[u_\theta] \in C([0,T] \times [0,1])$ for every $\theta \in \Theta$.

- Spatial boundary Residual given by,

$$\begin{aligned}
r_{sb1}[u_\theta](0,t) &:= u_\theta(0,t) - h_1(t), \quad \forall t \in (0,T), \\
r_{sb2}[u_\theta](1,t) &:= u_\theta(1,t) - h_2(t), \quad \forall t \in (0,T), \\
r_{sb3}[u_\theta](0,t) &:= (u_\theta)_x(0,t) - h_3(t), \quad \forall t \in (0,T), \\
r_{sb4}[u_\theta](1,t) &:= (u_\theta)_x(1,t) - h_4(t), \quad \forall t \in (0,T), \\
r_{sb5}[u_\theta](1,t) &:= (u_\theta)_{xx}(1,t) - h_5(t), \quad \forall t \in (0,T).
\end{aligned} \tag{3.68}$$

  Given the fact that the neural network and boundary data are smooth, above residuals are well-defined.

- Temporal boundary Residual given by,

$$r_{tb}[u_\theta](x) := u_\theta(x,0) - \bar{u}(x), \quad \forall x \in (0,1). \tag{3.69}$$

  Again the above quantity is well-defined and $r_{tb}[u_\theta] \in C^5((0,1))$, as both the initial data and the neural network are smooth.

Loss function

We set the following loss function

$$J(\theta) := \sum_{n=1}^{N_{tb}} w_n^{tb} |r_{tb}[u_\theta](x_n)|^2 + \sum_{n=1}^{N_{sb}} \sum_{i=1}^{5} w_n^{sb} |r_{sbi}[u_\theta](t_n)|^2 + \lambda \sum_{n=1}^{N_{int}} w_n^{int} |r_{int}[u_\theta](x_n, t_n)|^2. \tag{3.70}$$

Here the residuals are defined by (3.69), (3.68), (3.67), w_n^{tb} are the N_{tb} quadrature weights corresponding to the temporal boundary training points $\mathcal{S}_{tb}$, w_n^{sb} are the N_{sb} quadrature weights corresponding to the spatial boundary training points $\mathcal{S}_{sb}$ and w_n^{int} are the N_{int} quadrature weights corresponding to the interior training points $\mathcal{S}_{int}$. Furthermore, λ is a hyperparameter for balancing the residuals, on account of the PDE and the initial and boundary data, respectively.

3.6.3 Estimate on the generalization error

We are interested in estimating the following generalization error for the PINN $u^* = u_{\theta^*}$ with loss function (3.70), for approximating the solution of (3.66):

$$\mathcal{E}_G := \left(\int_0^T \int_0^1 |u(x,t) - u^*(x,t)|^2 dx dt \right)^{\frac{1}{2}} . \tag{3.71}$$

We are going to estimate the generalization error in terms of the *training error* that we define as,

$$\mathcal{E}_T^2 := \underbrace{\sum_{n=1}^{N_{tb}} w_n^{tb} |r_{tb}[u_{\theta^*}](x_n)|^2}_{(\mathcal{E}_T^{tb})^2} + \underbrace{\sum_{n=1}^{N_{sb}} \sum_{i=1}^{5} w_n^{sb} |r_{sbi}[u_{\theta^*}](t_n)|^2}_{(\mathcal{E}_T^{sb})^2} + \lambda \underbrace{\sum_{n=1}^{N_{int}} w_n^{int} |r_{int}[u_{\theta^*}](x_n,t_n)|^2}_{(\mathcal{E}_T^{int})^2} . \tag{3.72}$$

Note that the training error can be readily computed *a posteriori* from the loss function (3.70).

We also need the following assumptions on the quadrature error. For any function $g \in C^k(\Omega)$, the quadrature rule corresponding to quadrature weights w_n^{tb} at points $x_n \in \mathcal{S}_{tb}$, with $1 \leq n \leq N_{tb}$, satisfies

$$\left| \int_\Omega g(x) dx - \sum_{n=1}^{N_{tb}} w_n^{tb} g(x_n) \right| \leq C_{quad}^{tb}(\|g\|_{C^k}) N_{tb}^{-\alpha_{tb}}. \tag{3.73}$$

For any function $g \in C^k(\partial\Omega \times [0,T])$, the quadrature rule corresponding to quadrature weights w_n^{sb} at points $(x_n, t_n) \in \mathcal{S}_{sb}$, with $1 \leq n \leq N_{sb}$, satisfies

$$\left| \int_0^T \int_{\partial\Omega} g(x,t) ds(x) dt - \sum_{n=1}^{N_{sb}} w_n^{sb} g(x_n,t_n) \right| \leq C_{quad}^{sb}(\|g\|_{C^k}) N_{sb}^{-\alpha_{sb}}. \tag{3.74}$$

Finally, for any function $g \in C^\ell(\Omega \times [0,T])$, the quadrature rule corresponding to quadrature weights w_n^{int} at points $(x_n, t_n) \in \mathcal{S}_{int}$, with $1 \leq n \leq N_{int}$, satisfies

$$\left| \int_0^T \int_\Omega g(x,t) dx dt - \sum_{n=1}^{N_{int}} w_n^{int} g(x_n,t_n) \right| \leq C_{quad}^{int}(\|g\|_{C^\ell}) N_{int}^{-\alpha_{int}}. \tag{3.75}$$

In the above, $\alpha_{int}, \alpha_{sb}, \alpha_{tb} > 0$ and in principle, different order quadrature rules can be used. We estimate the generalization error for the PINN in the following,

Theorem 3.6.2. *Let $u \in C^5([0,1] \times [0,T])$ be the unique classical solution of the Korteweg de-Vries & Kawahara equation (3.66). Let $u^* = u_{\theta^*}$ be a PINN generated by algorithm 1, corresponding to loss function (3.6), (3.70). Then the generalization error (3.71) can be estimated as,*

$$\begin{aligned}\mathcal{E}_G \leq C_1\big(&\mathcal{E}_T^{tb} + \mathcal{E}_T^{int} + C_2(\mathcal{E}_T^{sb}) + C_3(\mathcal{E}_T^{sb})^{1/2} \\ &+ (C_{quad}^{tb})^{1/2} N_{tb}^{-\alpha_{tb}/2} + (C_{quad}^{int})^{1/2} N_{int}^{-\alpha_{int}/2} + C_2(C_{quad}^{sb})^{1/2} N_{sb}^{-\alpha_{sb}/2} + C_3(C_{quad}^{sb})^{1/4} N_{sb}^{-\alpha_{sb}/4}\big),\end{aligned} \tag{3.76}$$

where

$$\begin{aligned}&C_1 = \sqrt{T + 2C_4 T^2 e^{2C_4 T}}, \quad C_2 = \sqrt{\|u\|_{C_t^0 C_x^0} + 1}, \\ &C_3 = \sqrt{10(\|u^*\|_{C_t^0 C_x^4} + \|u\|_{C_t^0 C_x^4})T^{1/2}}, \quad C_4 = \|u^*\|_{C_t^0 C_x^1} + \frac{1}{2}\|u\|_{C_t^0 C_x^1} + \frac{1}{2},\end{aligned} \tag{3.77}$$

and $C_{quad}^{tb} = C_{quad}^{tb}(\|r_{tb}[u_{\theta^*}]\|_{C^5})$, $C_{quad}^{sb} = C_{quad}^{sb}(\sum_{i=1}^{5} \|r_{sbi}[u_{\theta^*}]\|_{C^3})$ and $C_{quad}^{int} = C_{quad}^{int}(\|r_{int}[u_{\theta^*}]\|_{C^0})$ are the constants defined by the quadrature error (3.73), (3.74), (3.75), respectively.

The proof of theorem (3.6.2) is available in [64], which also contains further theoretical results on dispersive equations, such as the Benjamin-Ono and Camassa-Holm equations.

3.7 Numerical Experiments

In this section, we present numerical experiments for the approximation of solutions of the equations above by PINNs, generated with algorithm 1.

3.7.1 Ensemble Training

PINNs include several hyperparameters, including number of hidden layers K and neurons $\bar{d}$ of the networks, residual parameter λ, etc. A user is always confronted with the question of which parameter to choose. It is standard practice in machine learning to perform a systematic hyperparameter search. To this end, we follow the *ensemble training* procedure of [12]: for each of them, the model is retrained n_θ times with different starting values of the trainable weights in the optimization algorithm and the one resulting in the smallest value of the training loss is selected.

3.7.2 Semi-linear Parabolic Equation

For the first numerical experiment, as an example of semi-linear parabolic equation, we consider the heat equation in one space dimension i.e, $d = 1$ in (3.17) with $f \equiv 0$, domain $D = [-1, 1]$, $T = 1$, and initial data $\bar{u}(x) = -\sin(\pi x)$. Then, the exact solution of the heat equation is given by,

$$u(x,t) = -\sin(\pi x)e^{-\pi^2 t}. \tag{3.78}$$

Clearly both the initial data and the exact solution are smooth and Theorem 3.3.1 holds. Our first aim in this experiment is to illustrate the estimate (3.28) on the generalization error. To this end, we choose training the training sets $\mathcal{S}_{int}, \mathcal{S}_{sb}, \mathcal{S}_{tb}$ as quadrature points corresponding to the composite midpoint rule 2.19. In this case, $\alpha_{int} = 1$, $\alpha_{tb} = \alpha_{sb} = 2$. We run algorithm 1 with the following hyperparameters: we consider a fully connected neural network architecture (2.24), with the tanh activation function, with

3 hidden layers and 10 neurons in each layer, resulting in neural networks with less than 500 tuning parameters. Moreover, we use the loss function (3.22), with $\lambda = 1$ and with $q = 2$ i.e L^2-regularization, with regularization parameter $\lambda_{reg} = 0$. Finally, the optimizer is the second-order LBFGS method.

On this hyperparameter configuration, we consider two cases: (1) we vary the number of training points N_{int} while keeping N_{tb} and N_{sb} constant $N_{tb} = N_{sb} = 800$ ($N_{int} = [4096, 16384]$) and (2) we vary the number of training points $N_{sb} = N_{tb} = [25, 50, 100, 200, 400, 800, 1600]$, while keeping constant $N_{int} = 1024$. We then run the algorithm 1 to obtain the corresponding trained neural network and evaluate the resulting errors i.e, the training errors (3.24), generalization error (3.23), and the upper bound in (3.28). The generalization error (3.23) is computed by evaluating the error of the neural network with respect to the exact solution (3.78) on a *randomly chosen test set* of 10^5 points. Since the results are very susceptible to the initialization of the network parameters, we retrain each model 10 times, every time with different initial weights, and compute the averages of the above mentioned quantities. The results for this procedure are shown in figure 3.1. We see from this figure that the generalization error (3.23) is very low to begin with and decays with both the number of interior training points N_{int} and $N_u = N_{sb} + N_{tb}$. On the other hand, while the computable upper bound (3.28) decays with respect to increasing the number of interior training points, it is approximately constant with respect to increasing number of boundaries points. This can be explained in terms of the very low values of C^{sb}_{quad}, C^{tb}_{quad}, compared to C^{int}_{quad}. In both cases, the plot shows that the constants in the bound (3.23) are not blowing up as the number of training points is increased, and, more importantly, both the computed generalization error and the upper bound follow the same decay in the number of training samples. However, this upper bound does appear to be a significant overestimate as it is almost three orders of magnitude greater than the actual generalization error. This is not surprising as we had used non-sharp estimates such as triangle inequality and Grönwall's inequality rather indiscriminately while deriving (3.28). The plots also show that the training errors (3.24) are slightly larger than the computed generalization errors for this example, in particular for large number of training samples. Note that this observation is still consistent with the bound (3.28). Given the fact that the training error is defined in terms of residuals and the generalization error is the error in approximating the solution of the underlying PDE by the PINN, there is no reason, a priori, to expect that the generalization error should be greater than the training error.

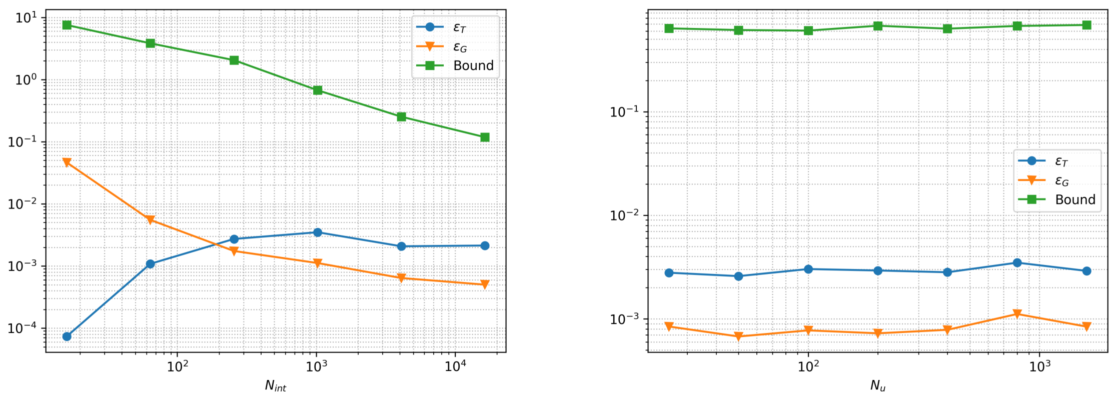


Figure 3.1: Generalization error, training error and theoretical bound (3.28) VS number of training samples N_{int} (left) and number of training samples $N_u = N_{sb} + N_{tb}$

	$K-1$	$\bar{d}$	q	λ_{reg}	λ
1D Heat Equation	2, 4, 8	12, 16, 20	1, 2	0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}	0.01, 0.1, 1, 10
Burgers Equation	4, 8, 10	16, 20, 24	2	0, 10^{-6}, 10^{-5}	0.1, 1, 10
Euler Equations, Taylor Vortex	4, 8, 12	16, 20, 24	1, 2	0, 10^{-6}, 10^{-5}	0.1, 1, 10
Euler Equations, Double Shear Layer	16, 20, 24	32, 40, 48	1, 2	0, 10^{-6}, 10^{-5}	0.1, 1, 10
KdV Equation	4, 8	20, 24, 28	2	0	0.1, 1, 10
Kawahara Equation	4, 8, 12	20, 24, 28, 32	2	0	0.1, 1, 10

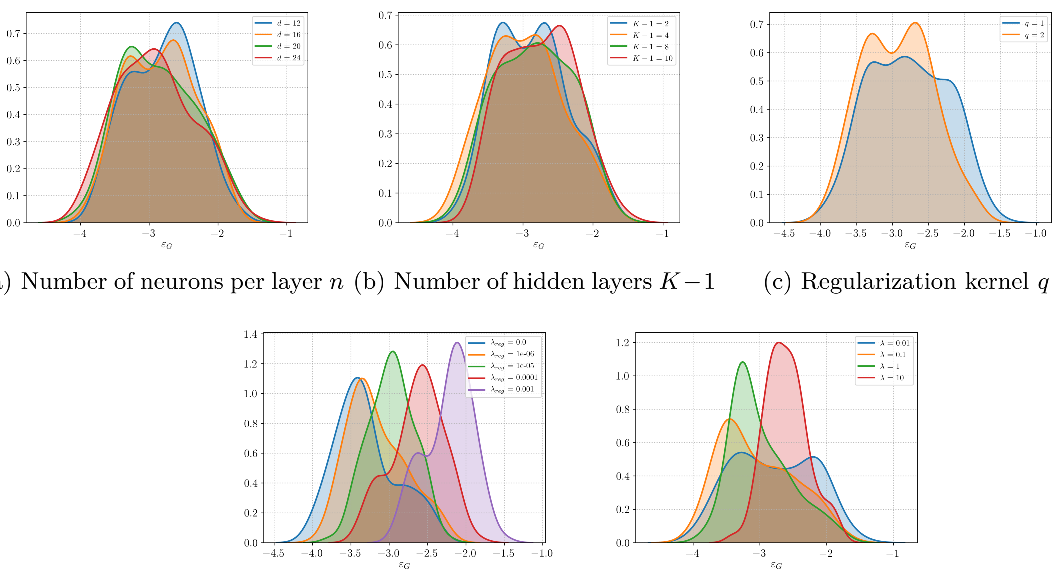
Table 3.1: Hyperparameter configurations employed in the ensemble training of PINNs: $K-1$ is the number of hidden layers, $\bar{d}$ is the number of neurons per layer, q and λ_{reg} are the exponent in the regularization term (3.8) and the regularization parameter, respectively, λ is the scalar balancing the role of PDE and the data loss.

Sensitivity to Hyperparameters

A PINN involves several hyperparameters, some of which are shown in Table 3.1. An user is always confronted with the question of which parameter to choose. The theory, presented in this paper and in the literature, offers very little guidance about the choice of hyperparameters. Instead, it is standard practice in machine learning to do a systematic hyperparameter search. To this end, we follow the *ensemble training* procedure of [12] with a randomly chosen training set ($N_{int} = 1024, N_{sb} = N_{tb} = 64$) and compute the marginal distribution of the generalization error with respect to different choices of the hyperparameters (see Table 3.1). The ensemble training results in a total number of 360 configurations. For each of them, the model is retrained five times with different starting values of the trainable weights in the optimization algorithm and the one resulting in the smallest value of the training loss is selected. We plot the corresponding histograms, visualizing the marginal generalization error distributions, in figure 3.2.

As seen from figure 3.2, there is a large variation in the spread of the generalization error, often two to three orders of magnitude, indicating sensitivity to hyperparameters. However, even the worst case errors are fairly low for this example. Comparing different hyperparameters, we see that there is not much sensitivity to the network architecture (number of hidden layers and number of neurons per layer) and a slight regularization or no regularization in the loss function (3.6) is preferable to large regularizations. The most sensitive parameter is λ in (3.22) where $\lambda = 1$ or 0.1 are significantly better than larger values of λ. This can be explained in terms of the bound (3.28), where the boundary residual r_{sb} has a larger weight in error. A smaller value of λ enforces this component of error, and hence the overall error, to be small, and we see exactly this behavior in the results.

Finally, in figure 3.3, we plot the total training error (3.24) on a logarithmic scale (x-axis) against the generalization error (3.23) (in log scale) (y-axis) for all the hyperparameter configurations in the ensemble training. This plot clearly shows that the two errors are highly correlated and validates the fundamental point of the estimates (3.28) and (3.28) that if the PINNs are trained well, they generalize very well. In other words, low training errors imply low generalization errors. Moreover, we see from figure 3.3 that the generalization error scales as a square root of the total training error, as predicted by the estimate (3.28).



(a) Number of neurons per layer n (b) Number of hidden layers $K-1$ (c) Regularization kernel q

(d) Regularization parameter λ_{reg} (e) Residual parameter λ

Figure 3.2: Marginal distributions of the (log) generalization error to different network hyperparameters

3.7.3 Viscous Scalar Conservation Law

We consider the viscous scalar conservation law (3.32), but in the domain $D_T = [-1,1] \times [0,1]$, with initial conditions, $\bar{u}(x) = -\sin(\pi x)$ and zero Dirichlet boundary conditions. We choose the flux function $f(u) = \frac{u^2}{2}$, resulting in the well-known *viscous Burgers'* equation.

This problem is considered for 4 different values of the viscosity parameter $\nu = \frac{c}{\pi}$, with $c = 0.01$, 0.005, 0.001, 0.0, and with $N_{int} = 8192$, $N_{tb} = 256$, $N_{sb} = 256$ points. All the training points are chosen as low-discrepancy Sobol sequences on the underlying domains. An ensemble training procedure, based on the hyperparameters presented in Table 3.1, is performed and the best performing hyperparameters i.e. those that led to the smallest training errors, are chosen and presented in Table 3.2.

In figure 3.4, we present the *reference* solution field $u(\cdot,t)$, at different time snapshots, of the viscous Burgers' equation computed with a simple upwind finite volume scheme and forward Euler time integration with 2×10^6 Cartesian grid points in space-time, and the predicted solution $u^*(\cdot,t)$ of the PINN, generated with algorithm 1, corresponding to the best performing hyperparameters (see Table 3.2), for different values of the viscosity coefficient. From this figure, we observe that for the viscosity coefficients, corresponding to $c = 0.01, 0.005$, the approximate solution, predicted by the PINN, approximates the underlying exact solution, which involves self steepening of the initial sine wave into a steady sharp profile at the origin, very well. This is further reinforced by the very low (relative percentage) generalization errors of approximately 1%, presented in Table 3.2. However, this efficient approximation is no longer the case for the inviscid problem i.e $\nu = 0$. As seen from figure 3.4 (bottom right), the PINN fails to resolve the solution, which in this case, consists of a steady shock at the origin. In fact, this failure to approximate is already seen with a viscosity coefficient of $\nu = \frac{0.001}{\pi}$. For this very low viscosity coefficient, we see that the relative generalization error is approximately 11%. The generalization error rises to 23% for the inviscid Burgers' equation. This increase in error appears consistent with the bound (3.39), combined

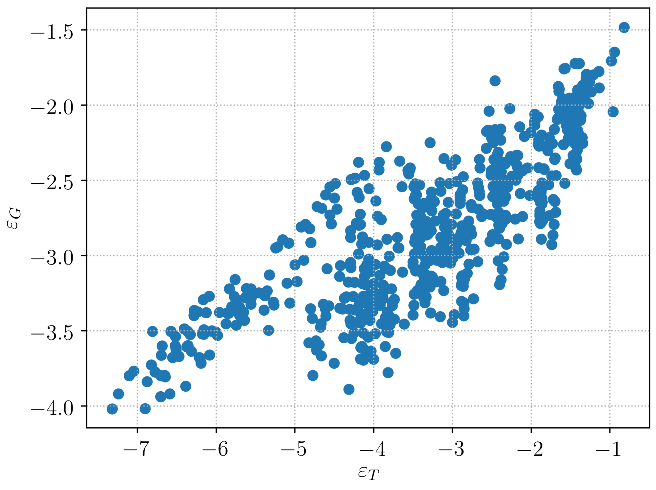


Figure 3.3: Log of Training error (X-axis) vs Log of Generalization error (Y-axis) for each hyperparameter configuration during ensemble training.

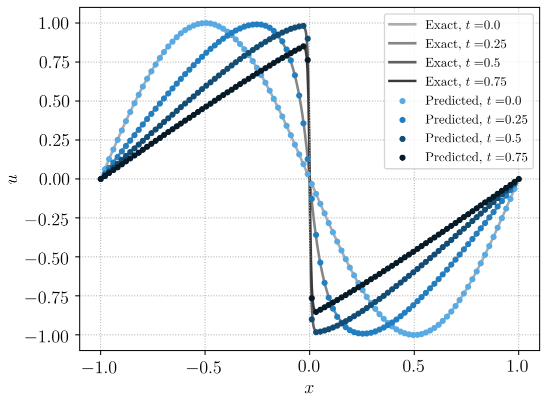
with the blow-up estimate (3.50) for the derivatives of the viscous Burgers' equation. As the viscosity $\nu \to 0$, the bounds in the right-hand-side of (3.39) can increase exponentially, which appears to be the case here.

To further test the bound (3.39)'s ability to explain performance of PINNs for the viscous Burgers' equation, we consider the following initial and boundary conditions,

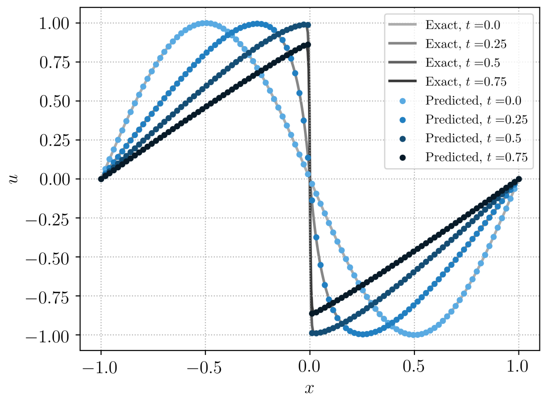
$$\bar{u}(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases} \quad \forall\, x \in [-1, 1],$$

$$u(t, -1) = 0,\ u(t, 1) = 1,\ \forall\, t \in [0, 1]. \tag{3.79}$$

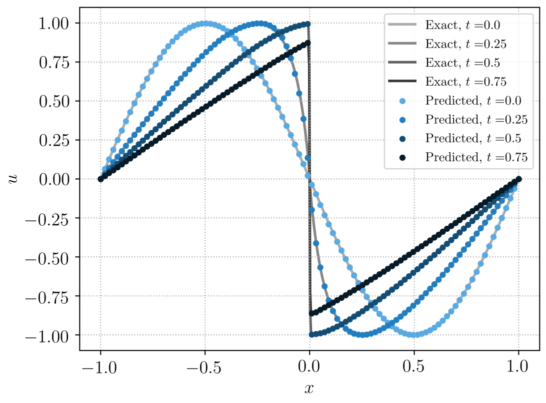
Given the discontinuity in the initial data we train the PINNs with a larger number of boundary training samples $N_{tb} = 512$ and $N_{sb} = 512$, while leaving $N_{int} = 8192$, unchanged. As in the previous experiments, training sets are Sobol sequences and an ensemble training is preformed to configure the network architecture. The results are summarized in Table 3.3 and figure 3.5. In this case, the exact solution is a so-called rarefaction wave (see figure 3.5 for the reference solution, computed in the manner, analogous to the previous numerical experiment) and the gradient of the solution remains uniformly bounded, as the viscosity coefficient $\nu \to 0$. Hence, from the bound (3.39), we expect that PINNs will efficiently approximate the underlying solution for all values of the viscosity coefficient. This is indeed verified in the solution snapshots, presented figure 3.5, where we observe that the PINN approximates the reference solution quite well, for all values of the viscosity coefficient. This behavior is further verified in table 3.3, where we see that the generalization error (3.37), remains low (less than 2%) for all the values of viscosity and in fact, reduces slightly as $\nu \to 0$, completely validating the error estimate (3.39).



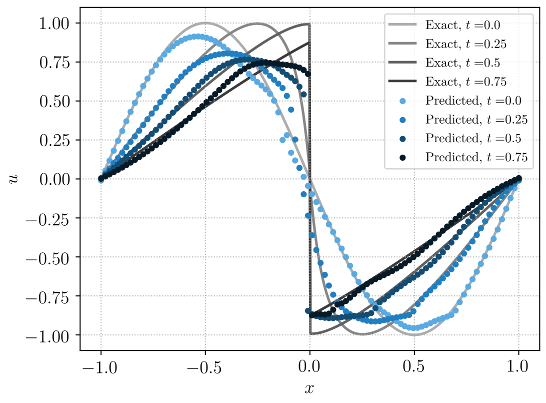
(a) $\nu = \frac{0.01}{\pi}$, $\mathcal{E}_G^r = 0.010$



(b) $\nu = \frac{0.005}{\pi}$, $\mathcal{E}_G^r = 0.012$



(c) $\nu = \frac{0.001}{\pi}$, $\mathcal{E}_G^r = 0.11$



(d) $\nu = 0$, $\mathcal{E}_G^r = 0.23$

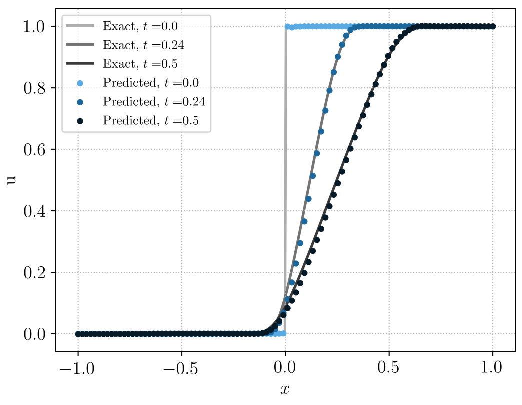
Figure 3.4: Burgers equation with discontinuos solution for different values of ν

3.7.4 Incompressible Euler Equations

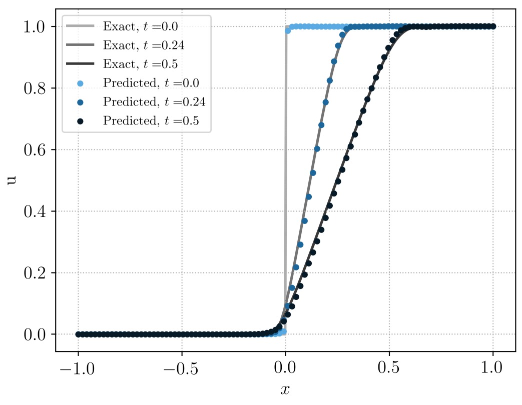
We present experiments for the incompressible Euler equations in two space dimensions, i.e $d = 2$ in (3.51). Moreover, we will use the *CELU* function, given by,

$$CELU(x) = \max(0, x) + \min\big(0, \exp(x) - 1\big), \tag{3.80}$$

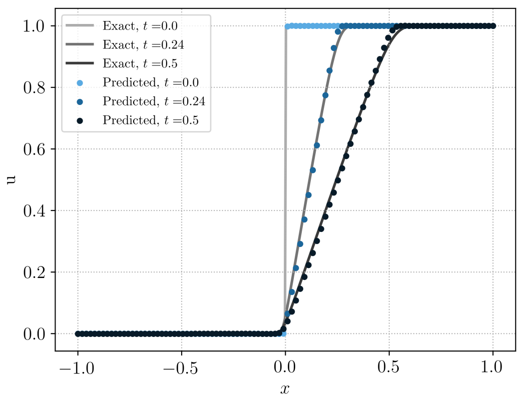
as the activation function σ in (2.24), [65]. The CELU function results in better approximation than the hyperbolic tangent, for the Euler equations.



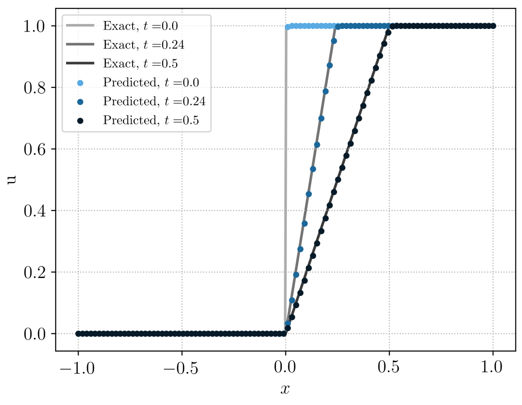
(a) $\nu = \frac{0.01}{\pi}$, $\mathcal{E}_G^r = 0.022$



(b) $\nu = \frac{0.005}{\pi}$, $\mathcal{E}_G^r = 0.018$



(c) $\nu = \frac{0.001}{\pi}$, $\mathcal{E}_G^r = 0.016$



(d) $\nu = 0$, $\mathcal{E}_G^r = 0.012$

Figure 3.5: Burgers equation with rarefaction wave for different values of ν

Taylor Vortex

In the first numerical experiment, we consider the well-known Taylor Vortex, in a computational domain $D_T = [-8, 8]^2 \times [0, 1]$ with periodic boundary conditions and with the initial conditions,

$$\begin{aligned} \bar{u}(x,y) &= -ye^{\frac{1}{2}(1-x^2-y^2)} + a_x, \qquad (x,y) \in [-8,8]^2, \\ \bar{v}(x,y) &= xe^{\frac{1}{2}(1-x^2-y^2)} + a_y, \qquad (x,y) \in [-8,8]^2, \end{aligned} \tag{3.81}$$

with $a_x = 4$ and $a_y = 0$.

In this case, one can obtain the following exact solution,

$$\begin{aligned} u(x,y,t) &= -(y - a_y t)e^{\frac{1}{2}\left[1-(x-a_x t)^2-(y-a_y t)^2\right]} + a_x, \\ v(x,y,t) &= (x - a_x t)e^{\frac{1}{2}\left[1-(x-a_x t)^2-(y-a_y t)^2\right]} + a_y. \end{aligned} \tag{3.82}$$

ν	N_{int}	N_{sb}	N_{tb}	$K-1$	$\bar{d}$	L^1-reg.	L^2-reg.	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$
$0.01/\pi$	8192	256	256	8	20	0.0	0.0	0.1	0.0005	1.0%
$0.005/\pi$	8192	256	256	10	20	0.0	0.0	0.1	0.00075	1.2%
$0.001/\pi$	8192	256	256	10	20	0.0	10^{-6}	0.1	0.009	11.0%
0.0	8192	256	256	8	24	0.0	10^{-5}	0.1	0.08	23.0%

Table 3.2: Best performing *Neural Network* configurations for the Burgers equation with shock, for different values of the parameter ν.

ν	N_{int}	N_{sb}	N_{tb}	$K-1$	$\bar{d}$	L^1-reg.	L^2-reg.	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$
$0.01/\pi$	8192	512	512	4	20	0.0	0.0	0.1	0.0043	2.2%
$0.005/\pi$	8192	512	512	4	20	0.0	0.0	0.1	0.0034	1.8%
$0.001/\pi$	8192	512	512	4	16	0.0	0.0	0.1	0.00048	1.6%
0.0	8192	512	512	4	20	0.0	0.0	0.1	0.00033	1.2%

Table 3.3: Best performing *Neural Network* configurations for the Burgers equation with rarefaction wave, for different values of the parameter ν.

We will generate the training set with $N_{int} = 8192$, $N_{tb} = N_{sb} = 256$ points, chosen as low-discrepancy Sobol sequences on the underlying domains. An ensemble training procedure is performed, as described in the previous section, and resulted in the hyperparameter configuration presented in Table 3.4.

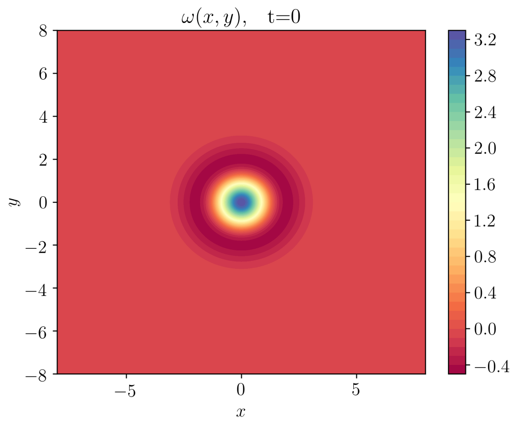
To visualize the solution, we follow standard practice and compute the vorticity $\omega = \mathrm{curl}(u)$ and present the exact vorticity and the one obtained from the PINN, generated by algorithm 1 in figure 3.6. We remark that the vorticity can be readily computed from the PINN u^* by automatic differentiation. We see from the figure, that the PINN, approximates the flow field very well, both initially as well as at later times, with small numerical errors. This good quality of approximation is further reinforced by the generalization error (3.57), computed from (3.82) with 10^5 uniformly distributed random points, and presented in Table 3.4. We see that the generalization error for the best hyperparameter configuration is only 0.012%, indicating very high accuracy of the approximation for this test problem.

	N_{int}	N_{sb}	N_{tb}	$K-1$	$\bar{d}$	L^1-reg.	L^2-reg.	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$
Taylor Vortex	8192	256	256	12	24	0.0	0.0	1	0.0003	0.012%
Double Shear Layer	65536	16384	16384	24	48	0.0	0.0	0.1	0.0025	3.8%

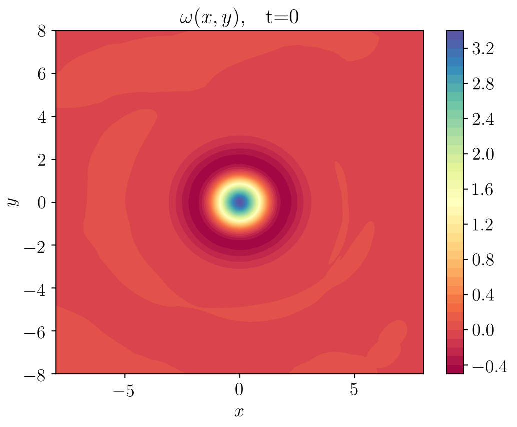
Table 3.4: Best performing *Neural Network* configurations for the Taylor Vortex and Double Shear Layer problem. Low-discrepancy Sobol points are used for every reported numerical example.

Double shear Layer

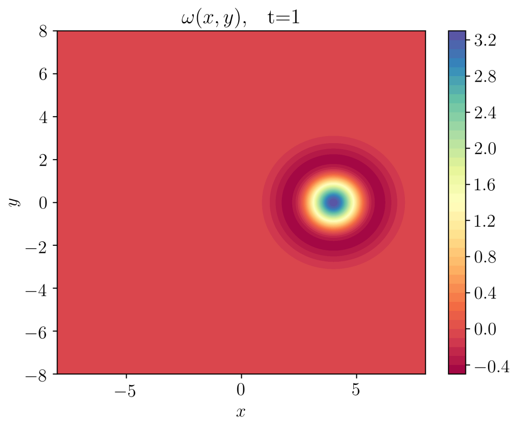
We consider the two-dimensional Euler equations (3.51) in the spatial computational domain $D = [0, 2\pi]^2$ with periodic boundary conditions and consider initial data with the underlying vorticity, shown in figure 3.7 (Top Left). This vorticity, corresponds to a velocity field that has been evolved with a standard



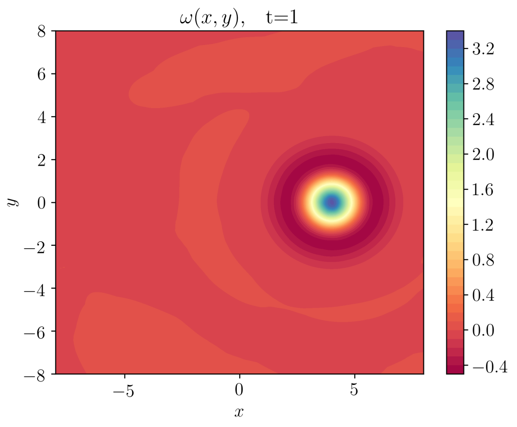
(a) Exact vorticity at $T = 0$



(b) Approximate (PINN) vorticity at $T = 0$



(c) Exact vorticity at $T = 1$



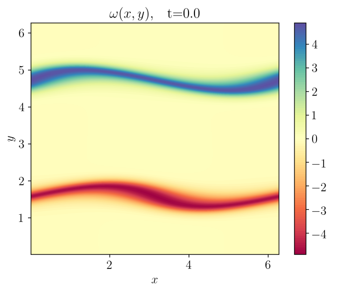
(d) Approximate (PINN) vorticity at $T = 1$

Figure 3.6: Exact and PINN solutions to the Taylor Vortex

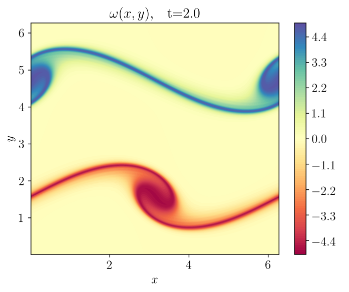
second-order finite difference projection method, with the well-known double shear layer initial data [66], evolved until $T = 1$. We are interested in determining if we can train a PINN to match the solution for later times.

To this end, we acknowledge that the underlying solution is rather complicated (see figure 3.7 Top row) for the corresponding reference vorticity, and consists of fast moving sharp vortices. Moreover, the vorticity is high, implying from the bound (3.59), that the generalization errors with PINNs can be high in this case. Hence, we consider training sets with larger number of points than the previous experiment, by setting $N_{int} = 65536$ and $N_{tb} = N_{sb} = 16384$. The ensemble training procedure resulted in hyperparameters presented in Table 3.4.

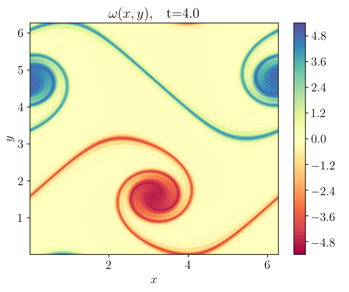
We present the approximate vorticity computed with the PINN, together with the exact vorticity, in figure 3.7, at three different times. From the figure, we see that the vorticity is approximated by the PINN quite well. However, the sharp vortices are smeared out and this is particularly apparent at later times. This is not surprising as the underlying solution is much more complicated in this case. Moreover, we have trained the PINN to approximate the velocity field, rather than the vorticity, and the generalization



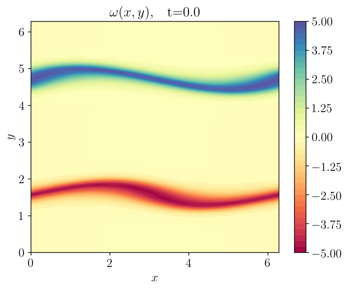
(a) Reference, $T = 0$



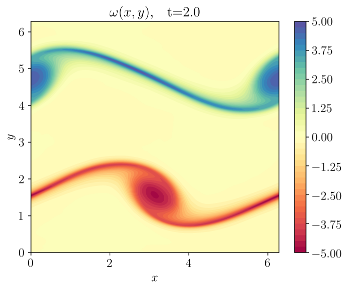
(b) Reference, $T = 2$



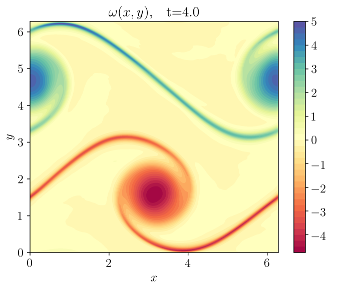
(c) Reference, $T = 4$



(d) PINN, $T = 0$



(e) PINN, $T = 2$



(f) PINN, $T = 4$

Figure 3.7: Reference (Top Row) and PINN generated (Bottom Row) vorticities for the double shear layer problem at different times

error (3.57) is still quite low at 3.8% (see Table 3.4).

3.7.5 KdV equation

We set $\beta = 0$ in (3.17) to recover the KdV equation and consider the well-known numerical benchmarks of single and double soliton solutions, with exact solution formulas for both cases.

For the single soliton, the exact solution is given by,

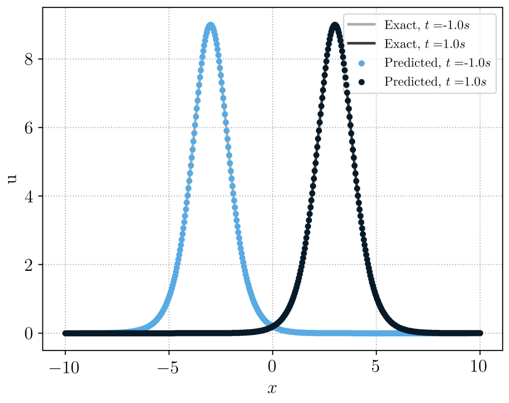
$$u(x,t) = 9\text{sech}^2(\sqrt{3/4}(x-3t)), \tag{3.83}$$

representing a single bump moving to the right with speed 3 with initial peak at $x = 0$.

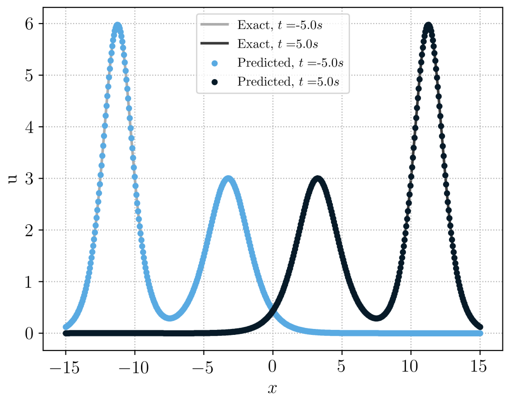
The ensemble training for the PINNs in this case resulted in the selection of hyperparameters, reported in Table 3.5. We plot the exact solution and the approximate solution, computed with the PINNs algorithm 1 in figure 3.8 (left). As seen from this figure, PINNs provide a very accurate approximation for the single soliton. This is further verified in the extremely low generalization errors reported in Table 3.5, showcasing the ability of PINNs to accurately approximate single solitons for the KdV equation.

The double soliton is instead described by the following equation

$$u(x,t) = 6(b-a)\frac{b\text{csch}^2(\sqrt{b/2}(x-2bt)) + a\text{sech}^2(\sqrt{a/2}(x-2at))}{\left(\sqrt{a}\tan(\sqrt{a/2}(x-2at)) - \sqrt{b}\tanh(\sqrt{b/2}(x-2bt))\right)^2}, \tag{3.84}$$



(a) Single soliton



(b) Double soliton

Figure 3.8: The exact and PINN solutions of single and double soliton test case of KdV equation.

	N_{int}	N_{sb}	N_{tb}	$K-1$	$\bar{d}$	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$
Single Soliton	2048	512	512	4	20	0.1	0.000236	0.00338%
Double Soliton	4096	1024	1024	4	32	1	0.000713	0.059%

Table 3.5: Best performing *Neural Network* configurations for the single soliton and double soliton problem. Low-discrepancy Sobol points are used for every reported numerical example.

where we set $a = 0.5$ and $b = 1$. Equation 3.84 represents two solitary waves which collide at $t = 0$ and separate for $t > 0$. For large $|t|$, $u(\cdot, t)$ is close to a sum of two single solitons at different locations.

The ensemble training for the PINNs in this case resulted in the selection of hyperparameters, reported in Table 3.5 (bottom row). We plot the exact solution and the approximate solution, computed with the PINNs algorithm 1 in figure 3.8 (right). As seen from this figure, PINNs provide a very accurate approximation for the double soliton, which is further verified in the extremely low generalization errors reported in Table 3.5. Thus, PINNs are able to approximate KdV solitons to very high accuracy.

Iterations	Training Time[s]	$\mathcal{E}_T$	$\mathcal{E}_G^r$
100	4	6.75e-02	1.84e-01
500	21	2.41e-03	1.65e-03
1000	44	7.34e-04	4.92e-04
2000	61	2.36e-04	3.38e-05

Table 3.6: Results of different training iterations for single soliton case of KdV equation.

We further investigate the computational cost of PINNs in approximating the KdV solutions. In particular, we consider the training cost, quantified in terms of the number of LBFGS iterations and the corresponding elapsed time, as the main indicators of the total computational burden of algorithm 1. Tables 3.6 and 3.7 provide detailed information on the training times (in seconds) for different numbers of iterations, as well as the achieved errors (both training and generalization errors) for the single and double soliton test cases, respectively. Upon analyzing the data in Table 3.6, we observe that training the PINN for approximating the single soliton is remarkably fast. A relative error of 1% is already achieved with fewer than 500 LBFGS iterations, requiring approximately 20 seconds of training time.

In contrast, training the PINN to accurately represent the double soliton takes a longer duration. Achieving an error of 1% necessitates approximately 2000 iterations, with a training time of under 3 minutes. This outcome is not surprising, considering the significantly more complicated structure of the double soliton. Nevertheless, despite the increased training time, the overall computational cost remains quite low, considering the high accuracy achieved.

Iterations	Training Time[s]	$\mathcal{E}_T$	$\mathcal{E}_G^r$
100	9	1.21e-01	4.82e-01
500	48	2.60e-02	1.30e-01
1000	95	7.00e-03	4.32e-02
2000	159	2.54e-03	1.11e-02
5000	436	7.89e-04	6.50e-04
10000	499	7.13e-04	5.88e-04

Table 3.7: Results of different training iterations for double soliton case of KdV equation.

Solving the Parametrized KdV Equation.

As highlighted in the introduction, a key advantage of deep learning in scientific computing is the ability of neural networks to approximate solutions of high-dimensional PDEs at high accuracy.

As an example, we consider the following *parameterized* initial-value problem for the KdV equations:

$$\begin{aligned} u_t + \gamma u u_x + \kappa u_{xxx} &= 0, \\ \bar{u}(x,\alpha,\beta,\gamma) &= \frac{\beta}{\gamma} + \frac{\alpha-\beta}{\gamma}\text{sech}^2\Big(\sqrt{\frac{\alpha-\beta}{12\kappa}}(x)\Big) \end{aligned} \tag{3.85}$$

Here, α, β, γ are scalar parameters that specify the initial location and amplitude for the *soliton* initial data, and κ is a scalar parameter that measures the dispersivity of the medium.

We aim to obtain a solution of the PDE as a function of the time-space coordinate and the parameters $\alpha, \beta, \gamma, \kappa$ as well, i.e., $u = u(x, t, \alpha, \beta, \gamma, \kappa) : \Omega \subseteq \mathbb{R}^6 \to \mathbb{R}$. The solution accounts for a total of six dimensions. Hence, we classify this parameterized partial differential equation as a high-dimensional PDE.

It turns out that this parametrized KdV equation (3.85) admits an exact soliton solution given by:

$$u = \frac{\beta}{\gamma} + \frac{\alpha-\beta}{\gamma}\text{sech}^2\Big(\sqrt{\frac{\alpha-\beta}{12\kappa}}(x - (\beta + \frac{\alpha-\beta}{3})t)\Big) \tag{3.86}$$

We can readily see that the KdV single soliton solution (3.83) is recovered by setting $(\alpha, \beta, \gamma, \kappa) = (9, 0, 1, 1)$.

The solution is approximated with PINNs by collocating the PINN residual resulting from (3.85) on Sobol points from the underlying 6-dimensional domain. In particular, we choose $\alpha \sim \mathcal{U}(8.7, 9.3)$, $\beta \sim \mathcal{U}(-0.4, 0.4)$, $\gamma \sim \mathcal{U}(0.9, 1.1)$, and $\kappa \sim \mathcal{U}(0.9, 1.1)$ such that $\mathbb{E}(\alpha, \beta, \gamma, \kappa) = (9, 0, 1, 1)$. The initial and periodic boundary residuals are computed analogously.

Once the model is trained, we employ it to solve the *uncertainty quantification problem* as an example of a many-query problem. Specifically, we are interested in computing the mean and standard deviation of the pushforward measure of the solution of the PDE given the distributions of the input parameters mentioned above.

In Figure 3.9, we plot the mean $\pm$ standard deviation for both the initial data and the uncertain solution at a later time, and compare it with the exact solution computed from (3.86). From this figure, we observe that the statistical quantities computed with the PINN approximate the exact solution quite well. This qualitative observation is reinforced by the quantitative results presented in Table 3.8, where the generalization error, defined analogously to (3.23) by integrating over the parameter space, is observed to be less than 0.5% in approximately 5 minutes of training time. This result highlights the ability of PINNs to approximate *high-dimensional* parametric dispersive PDEs with high accuracy.

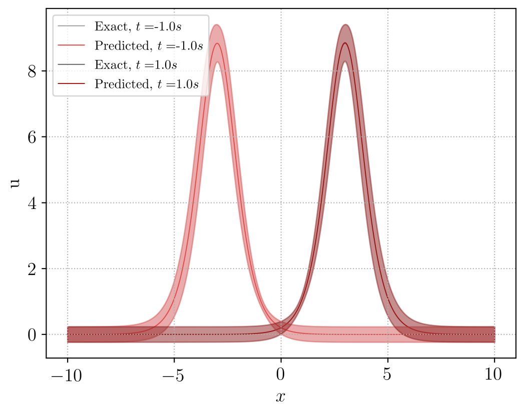


Figure 3.9: The mean and standard deviation plot of exact and PINN solution of parametrized single soliton test case of parametrized KdV equation (3.85).

	N_{int}	N_{sb}	N_{tb}	$K-1$	$\bar{d}$	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$
Single Soliton UQ	16384	4096	4096	4	24	0.1	0.00351	0.442%

Table 3.8: Best performing *Neural Network* configurations for the single soliton UQ test case for the parametrized KdV equations (3.85). Low-discrepancy Sobol points are used for every reported numerical example.

3.7.6 Kawahara equation

Following [67, 68, 69], we consider a Kawahara-type equation which differs from Kawahara equation (3.17) in a first-order term u_x,

$$u_t + u_x + uu_x + u_{xxx} - u_{xxxxx} = 0. \tag{3.87}$$

This first-order term u_x is a linear perturbation and we can easily derive a similar a posteriori bound on generalization error, as for (3.17). As no exact solution formulas for the double soliton test case are known for the Kawahara equation (3.87), we focus on the single soliton case, with exact solutions given by

$$u(x,t) = \frac{105}{169}\text{sech}^4\Big(\frac{1}{2\sqrt{13}}(x - \frac{205}{169}t - x_0)\Big). \tag{3.88}$$

This represents a single bump moving to the right with speed $\frac{205}{169}$ with initial peak at $x = x_0$. The ensemble training selected PINNs with hyperparameters, given in Table 3.9. The resulting PINN approximation, together with the exact solution is plotted in figure 3.10 and shows that the trained PINN approximates the exact solution with very high accuracy. This is further verified in the extremely low generalization error of 0.1%, reported in Table 3.9. In Table 3.10, we present training times (in seconds)

	N_{int}	N_{sb}	N_{tb}	$K-1$	$\bar{d}$	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$
Single Soliton	2048	512	512	4	24	10	0.000321	0.101%

Table 3.9: Best performing *Neural Network* configurations for the single soliton test case for the Kawahara equations (3.87). Low-discrepancy Sobol points are used for every reported numerical example.

for the PINNs algorithm for the Kawahara equation 3.87. We observe from this Table that an error of less than 1% percent is achieved in approximately $6-7$ minutes. Given the fact that the Kawahara equation requires the evaluation of 5-th order derivatives, it is expected that each training iteration is significantly more expensive than that of the KdV equation. Table 3.10 shows that this is indeed the case and partly explains the higher computational cost for the PINN to approximate the Kawahara equation. Nevertheless, the total cost is still considerably smaller than those reported for the finite difference scheme in [68, 69]. As an examples, to achieve 1% error, it takes approximately $15-18$ minutes for the dissipative finite-difference scheme presented in [69].

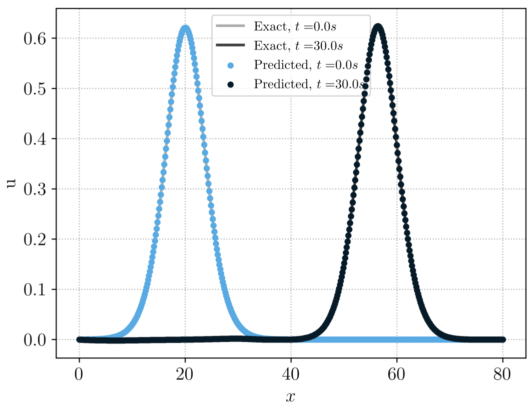


Figure 3.10: The exact and PINN solution of single soliton test case of Kawahara equation (3.87).

Iterations	Training Time$[s]$	$\mathcal{E}_T$	$\mathcal{E}_G^r$
100	25	8.89e-02	9.70e-01
500	127	4.76e-02	7.86e-01
1000	249	8.40e-03	1.89e-01
2000	466	1.06e-03	5.88e-03
5000	964	3.21e-04	1.01e-03

Table 3.10: Results of different training iterations for single soliton case of Kawahara equation.

4 wPINNs for the Forward Problem of Hyperbolic Conservation Laws

As shown in last chapter, the pointwise residuals associated with (approximations of) weak solutions can blow up. Hence, we need to replace these pointwise PDE residuals with suitable *weak* versions. This can be achieved by mimicking the weak formulation of the underlying conservation laws and integrating by parts with respect to smooth *test functions* to define a weak form of the PDE residual. Such weak versions of PINNs have already been considered in the context of so-called *variational PINNs* [70], where the test functions are selected as suitable basis functions, such as orthogonal polynomials. Such an approach can indeed be considered in our context. However, we refrain from doing so here as a key advantage for PINNs is that it is a *meshless* approach and does not require any underlying grid. Instead, we will leverage the universal approximation properties of neural networks and choose parametrized neural networks as our test functions. Neural networks are also used as approximations to the underlying solution i.e., as trial functions. Hence, our weak formulation leads to a *min-max* optimization problem where the neural network parameters (weights and biases) are maximized with respect to test functions and minimized with respect to trial functions.

However, working with the weak formulation alone does not suffice to build an accurate approximation strategy for scalar conservation laws as one also needs to incorporate entropy conditions. To this end, we will define a novel *entropy residual*, based on the well-known family of *Kruzkhov entropies* [23] and solve the corresponding min-max optimization problem for training the neural networks that will approximate the entropy solution accurately. We term the resulting construction as *wPINNs*.

4.1 Scalar Conservation Laws

Consider the scalar conservation law (3.32) with $\nu = 0$ and $D = [0, 1]$

$$\begin{aligned} u_t + f(u)_x = 0 \quad & x \in D,\ t \in [0,T] \\ u = g \quad & x \in \partial D,\ t \in [0,T] \\ u = \bar{u}(x) \quad & x \in D. \end{aligned} \tag{4.1}$$

Here, $u \in L^1(D \times (0,T))$ is the conserved quantity and f is the so-called flux function with $\bar{u}$ being the initial data. Moreover, the PDE (4.1) needs to be supplemented with suitable boundary conditions. We mostly consider periodic boundary conditions in this chapter.

Following [23], one defines *weak solutions* of (4.1) as follows,

Definition 4.1.1. *A function $u \in L^\infty(\mathbb{R} \times \mathbb{R}_+)$ is a weak solution of* (4.1) *with initial data $\bar{u} \in L^\infty(\mathbb{R})$ if*

$$\int_{\mathbb{R}_+} \int_{\mathbb{R}} (u\varphi_t + f(u)\varphi_x)\, dxdt + \int_{\mathbb{R}} \bar{u}(x)\varphi(x,0)dx = 0, \tag{4.2}$$

holds for all test functions $\varphi \in C_c^1(\mathbb{R} \times \mathbb{R}_+)$.

However, weak solutions are not unique [23]. To recover uniqueness, one needs to impose additional admissibility criteria or *entropy conditions*. To this end, we consider the so-called *Kruzkhov entropy functions*, given by $|u-c|$, for any $c \in \mathbb{R}$ and the resulting entropy flux functions,

$$\partial_t |u-c| + \partial_x Q[u;c] \leq 0 \qquad \text{where} \quad Q : \mathbb{R}^2 \to \mathbb{R} : (u,c) \mapsto Q(u,c) = \operatorname{sgn}(u-c)(f(u)-f(c)). \tag{4.3}$$

With this notation, we have the following definition of *entropy solutions*,

Definition 4.1.2. *We say that a function $u \in L^\infty(\mathbb{R} \times \mathbb{R}_+)$ is an entropy solution of (4.1) with initial data $\bar{u} \in L^\infty(\mathbb{R})$ if u is a weak solution of (4.1) and if u satisfies that*

$$\int_0^T \int_{\mathbb{R}} \left(|u-c|\, \varphi_t + Q[u;c]\varphi_x \right) dxdt - \int_{\mathbb{R}} \left(|u(x,T)-c|\, \varphi(x,T) - |\bar{u}(x)-c|\, \varphi(x,0) \right) dx \geq 0 \tag{4.4}$$

for all $\varphi \in C_c^1(\mathbb{R} \times \mathbb{R}_+)$, $c \in \mathbb{R}$ and $T > 0$.

It holds that entropy solutions are unique and continuous in time, as formulated below [23], where $\|\cdot\|_{TV}$ denotes the total variation seminorm.

Theorem 4.1.3. *Assume that $f \in C^1$ and $\bar{u} \in L^\infty \cap L^1$. Then there exists a unique entropy solution u of (4.1) and if $\|\bar{u}\|_{TV} < \infty$ then u satisfies the following,*

$$\|u(t)-u(s)\|_{L^1} \leq |t-s|\, M \|\bar{u}\|_{TV} \qquad \text{and} \qquad \|u(t)\|_{L^\infty} \leq \|\bar{u}\|_{L^\infty}, \ \|u(t)\|_{BV} \leq \|\bar{u}\|_{BV}, \tag{4.5}$$

where $M = M(\bar{u}) = \max_{essinf_x \bar{u}(x) \leq u \leq esssup_x \bar{u}(x)} |f'(u)|$.

4.2 Weak PINNs (*wPINNs*)

We will circumvent the failure of conventional PINNs that minimized the pointwise PDE residual

$$r_{int}[u_\theta](x,t) := \partial_t(u_\theta(x,t)) + \partial_x(f(u_\theta(x,t))) - \nu\partial_{xx}(u_\theta(x,t)), \quad x \in D, \ t \in [0,T] \tag{4.6}$$

defined in chapter 3, Section 3.4 by searching for neural networks that minimize a residual, related to the Kruzkhov entropy condition instead. To this end, we define for $v \in (L^\infty \cap L^1)(D \times [0,T])$, $\varphi \in W_0^{1,\infty}(D \times [0,T])$ and $c \in \mathbb{R}$ the following *Kruzhov entropy residual*,

$$\mathcal{R}(v,\varphi,c) := -\int_D \int_{[0,T]} \left(|v(x,t)-c|\, \partial_t\varphi(x,t) + Q[v(x,t);c]\partial_x\varphi(x,t) \right) dxdt. \tag{4.7}$$

Note that if u is an entropy solution of (4.1), then it holds that $\mathcal{R}(u,\varphi,c) \leq 0$. Similarly to standard PINNs, let us consider a feed-forward dense neural network u_θ (2.24) with tuning parameters $\theta \in \Theta$, approximating the entropy of the scalar conservation law (4.1):

$$u_\theta(x,t) \approx u(x,t), \quad x \in D, \ t \in [0,T] \tag{4.8}$$

and, motivated by (4.7), define the following residuals:

- *Interior residual* given by,

$$r_{int}[u_\theta, \varphi](x,t,c) := \partial_t \varphi(x,t)\,|u_\theta(x,t) - c| + Q[u_\theta(x,t); c]\partial_x \varphi(x,t). \tag{4.9}$$

  Observe that the residual depends on the test function $\varphi \in W_0^{1,\infty}$, and needs to be replaced by a finite-dimensional approximation. One possibility is to use locally supported (piecewise) polynomials or orthogonal polynomials such as Legendre polynomials. Such choices lead to what is often termed as variational PINNs. Instead, we defer from it and restrict the choice of the test function to the parametrized family of functions $\varphi_\eta(x,t)$ defined as $\varphi_\eta(x,t) = \omega(x,t)\xi_\eta(x,t)$. Here, $\omega : D \times [0,T] \to \mathbb{R}$ is a *cutoff* function satisfying the following properties:

  1. $\omega(x,t) = 1, \quad (x,t) \in D_\varepsilon^T,$
  2. $\omega(x,t) = 0, \quad (x,t) \in \partial(D \times [0,T]),$

$$D_\epsilon^T = \{(x,t) \in D \times [0,T] : dist((x,t), \partial(D \times [0,T])) < \epsilon\}, \tag{4.10}$$

  and $\xi_\eta(x,t)$ a neural network with trainable parameters η. The choice of the cutoff function guarantees that the test function has compact support. Then, the interior residual becomes,

$$r_{int}[u_\theta, \varphi_\eta] := \partial_t \varphi_\eta(x,t)\,|u_\theta(x,t) - c| + Q[u_\theta(x,t); c]\partial_x \varphi_\eta(x,t). \tag{4.11}$$

- *Spatial boundary residual* given by,

$$r_{sb}[u_\theta](x,t) := u_\theta(x,t) - g(x,t). \quad \forall x \in \partial D,\ t \in (0,T], \tag{4.12}$$

  Although the estimates below are derived assuming periodic boundary conditions, the numerical experiments are carried out with Dirichlet boundary conditions $g(x,t) = u|_{\partial D \times (0,T)}$.

- *Temporal boundary residual* given by,

$$r_{tb}[u_\theta](x) := u_\theta(x,0) - \bar{u}(x), \quad \forall x \in D. \tag{4.13}$$

Let us further define the following sets of training points:

- Interior collocation points $\mathcal{S}_{int} = \{y_m\}$ for $1 \le m \le N_{int}$, with each $y_m = (x_m, t_m) \in D \times [0,T]$.
- Spatial boundary collocation points $\mathcal{S}_{sb} = \{z_m\}$ for $1 \le m \le N_{sb}$ with each $z_m = (x_m, t_m)$ and $z_m \in \partial D \times [0,T]$.
- Temporal boundary collocation points $\mathcal{S}_{tb} = \{x_m\}$, with $1 \le m \le N_{tb}$ and $x_m \in D$.

The full set of collocation points is $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$ and $M = N_{int} + N_{sb} + N_{tb}$.

Given the definitions above, we consider the following loss function,

$$J(\theta, \eta, c) = J_{int}(\theta, \eta, c) + \lambda J_u(\theta) \tag{4.14}$$

with

$$J_{int}(\theta,\eta,c) = \frac{\left(ReLU\Big(-\sum_{m=1}^{N_{int}} r_{int}[u_\theta, \varphi_\eta](y_m, c)\Big)\right)^2}{\sum_{m=1}^{N_{int}} \varphi_\eta(y_m)^2 + \partial_x \varphi_\eta(y_m)^2}, \quad J_u(\theta) = \sum_{m=1}^{N_{tb}} r_{tb}[u_\theta](x_m)^2 + \sum_{m=1}^{N_{sb}} r_{sb}[u_\theta](z_m)^2. \tag{4.15}$$

Here, the denominator of J_{int} a Monte Carlo approximation of the H^1-norm of $\varphi_\eta(x,t)$. Eventually, we solve the min-max problem,

$$\theta^*, \eta^* c^* = \arg\min_{\theta \in \Theta} \max_{\eta \in \Theta} \max_{c \in \mathcal{C}} J(\theta, \eta, c), \tag{4.16}$$

and denote the corresponding NN u_{θ^*} as the *weak PINN*, *wPINN* for short.

Remark 4.2.1. *In practice, the maximization problem with respect to the scalar c is solved by computing $J_{max,C}(\theta,\eta) = \max_{c_i \in C} J_{max}(\theta,\eta,c_i)$, for a discrete set of values $C = \{c_i\}_{i=1}^{M}, c_i \in [c_{min}, c_{max}]$, whereas the optimization problems with respect to the neural network parameters θ and η is approximated with gradient descent and ascent, respectively.*

The algorithm for training *wPINN* is summarized below,

```
Algorithm 2: Training of wPINNs
Result: $\theta^*_{\mathcal{S}}, \eta^*_{\mathcal{S}}, c^*_{\mathcal{S}}$
Initialize the networks $u_\theta, \varphi_\eta : D \times [0,T] \to \mathbb{R}$ and $C$;
for $i = 1, ..., e$ do
    for $k = 1, ..., K_{max}$ do
        Compute $J_{max,C}(\theta,\eta) = \max_{c_i \in C} J_{max}(\theta,\eta,c_i)$;
        Update $\eta \leftarrow \eta + \tau_\eta \nabla J_{max,C}(\theta,\eta)$;
    end
    for $k = 1, ..., K_{min}$ do
        Compute $J_{max,C}(\theta,\eta) = \max_{c_i \in C} J_{max}(\theta,\eta,c_i)$;
        Update $\theta \leftarrow \theta - \tau_\theta \nabla(\lambda J_{max,C} + J_u)(\theta,\eta)$;
    end
end
```

4.3 Estimate of the Generalization Error

In this section, we will estimate the error due to the wPINN in approximating the entropy solution u of the scalar conservation law (4.1) with periodic boundary conditions $u(0,t) = u(1,t)$, $\forall t \in [0,T]$. The definition of the spatial boundary residual also has to be adjusted accordingly

$$r_{sb}[u_\theta](x,t) := u_\theta(0,t) - u_\theta(1,t), \quad \forall t \in (0,T]. \tag{4.17}$$

The relevant error in this context is the L^1-*error*:

$$\mathcal{E}_G = \mathcal{E}_G(\theta^*; \mathcal{S}) := \|u - u^*\|_{L^1} \tag{4.18}$$

As stated in section 3.2, the generalization error depends on the chosen training set $\mathcal{S}$ and the trained neural network with tuning parameters θ^*.

Note that there is no computation of the generalization error during the training process. On the other hand, we exclusively monitor the so-called *training error* given by,

$$\mathcal{E}_T := \mathcal{E}_T^{int} + \mathcal{E}_T^{sb} + \mathcal{E}_T^{sb} \tag{4.19}$$

with

$$\mathcal{E}_T^{int} := \left(\sum_{m=1}^{N_{int}} w_n |r_{int}[u_{\theta^*}, \varphi_{\eta^*}](y_m, c^*)|\right), \quad \mathcal{E}_T^{tb} := \left(\sum_{m=1}^{N_{tb}} w_m |r_{tb}[u_{\theta^*}](z_m)|\right) \quad \mathcal{E}_T^{sb} := \left(\sum_{m=1}^{N_{sb}} w_m |r_{sb}[u_{\theta^*}](x_m)|\right) \tag{4.20}$$

We further define the following set of test functions,

Definition 4.3.1. *Let for any $(y,s) \in [0,1] \times [0,T]$ and $\epsilon > 0$ the function $\overline{\varphi}_\epsilon^{y,s} : [0,1] \times [0,T] \to [0,\infty)$ be given by,*

$$\begin{aligned} \overline{\varphi}_\epsilon^{y,s}(x,t) &= \chi_\epsilon\left(\frac{t+s}{2}\right)\rho_\epsilon(x-y)\rho_\epsilon(t-s), \\ \chi_\epsilon(t) &= \frac{1}{2\sigma(\alpha\epsilon)}(\sigma(\alpha(t-2\epsilon)) - \sigma(\alpha(t-T+2\epsilon))), \qquad \alpha = 3\ln(1/\epsilon)/\epsilon, \\ \rho_\epsilon(x) &= \frac{\sigma(\beta(x+\epsilon^6)) - \sigma(\beta(x-\epsilon^6))}{2\epsilon^6}, \qquad \beta = 9\ln(1/\epsilon)/\epsilon^3, \end{aligned} \tag{4.21}$$

for $(x,t) \in [0,1] \times [0,T]$. Furthermore we define the set Φ_ϵ by,

$$\Phi_\epsilon = \left\{\overline{\varphi}_\epsilon^{y,s} \;:\; (y,s) \in [0,1] \times [0,T]\right\}. \tag{4.22}$$

Now, we will modify the famous doubling of variables argument of Kruzkhov to obtain the following bound on the L^1-error of *wPINNs* approximating the entropy solution of (4.1) with periodic boundary conditons,

Theorem 4.3.2. *Assume that u is the piecewise smooth entropy solution of* (4.1) *with essential range $\mathcal{C}$ and that $u(0,t) = u(1,t)$ for all $t \in [0,T]$. There is a constant $C > 0$ such that for every $\epsilon > 0$ and $v \in C^1(D \times [0,T])$, it holds that*

$$\begin{aligned} \int_0^1 |v(x,T) - u(x,T)|\, dx \le C\Bigg(&\int_0^1 |r_{tb}[v](x)|\, dx - \max_{c\in\mathcal{C},\varphi\in\Phi_\epsilon} \int_D \int_{[0,T]} r_{int}[v,\varphi](x,t,c) dx dt \\ &+ (1 + \|v\|_{C^1})\ln(1/\epsilon)^3\epsilon + \int_0^T |r_{sb}[v](t)|\, dt\Bigg). \end{aligned} \tag{4.23}$$

The proof of the theorem can be fund in [71]. The main point of Theorem 4.3.2 was to provide an upper bound on the L^1-generalization error in terms of the residuals. Once the stability estimated is derived, the bound of the generalization error in term of the training error can be obtained by simply observing that the training errors are quadrature approximation of the integrals in the stability estimate 4.23 similarly to what was established for the proofs presented in the previous chapter. Further details and theoretical results are reported in [71].

The following remarks are in order,

Remark 4.3.3. *Throughout the entire section, we have focused on calculating the L^1-error of the weak PINN, as the L^1-norm is a natural choice for scalar conservation laws. In practice, however, we observe that it is easier to train the weak PINN using an L^2-based loss function.*

Remark 4.3.4. *The terms appearing in the loss function* (4.15) *terms are approximations of the integrals in the error estimate, cf. Theorem 4.3.2, where the L^1 norm has been replaced by the L^2 (Remark 4.3.3). This is done to facilitate optimization of the resulting min-max problem.*

Remark 4.3.5. *We observe that additional terms, i.e., use of the ReLU function and test function H^1-seminorm, have been introduced in the definition of the loss function* (4.15)*, when compared to the terms in the error estimate in Theorem 4.3.2. These are introduced to facilitate training and the estimates can be readily extended to incorporate the contributions of these terms.*

Remark 4.3.6. *In experiments (cf. Section 4.2), one can replace $\mathcal{R}$ with the following alternative,*

$$\tilde{\mathcal{R}}(v,\varphi,c) := \int_D \int_{[0,T]} (\varphi(x,t)\partial_t |v(x,t)-c| - Q[v(x,t);c]\partial_x \varphi(x,t))\, dxdt. \tag{4.24}$$

The only difference with $\mathcal{R}$ is that in $\tilde{\mathcal{R}}$ the time derivative is with $|v(x,t)-c|$ and not with the test function. Because of Lemma A.6 in [71] it holds that $\left|\tilde{\mathcal{R}}(v,\varphi,c)-\mathcal{R}(v,\varphi,c)\right| = \mathcal{O}(\epsilon)$ if $\varphi \in \Phi_\epsilon$.

4.4 Implementation of *wPINNs*

To begin with, we describe some key implementation details.

4.4.1 Ensemble Training

wPINNs include several hyperparameters, including number of hidden layers L_θ, L_η, and neurons $\bar{d}_\theta$, $\bar{d}_\eta$ of the networks, the number of iterations K_{max}, K_{min}, number of epochs e, residual parameter λ, etc. A user is always confronted with the question of which parameter to choose. It is standard practice in machine learning to perform a systematic hyperparameter search. To this end, we follow the *ensemble training* procedure of [12]: for each configuration of the model hyperparameters we *retrain* the *wPINN* n_θ times, each with different initialisation of the networks hyperparameters, and select the hyperparameter configuration that minimises the average value over the retrainings of the following:

$$\begin{aligned}\mathcal{E}_T(\theta^*_{\mathbf{S}},\eta^*_{\mathbf{S}},c^*_{\mathbf{S}}) = \sum_{m=1}^{N_{int}} &(\varphi^*(y_m)\partial_t |u^*(y_m)-c^*_{\mathbf{S}}| - Q[u_\theta(y_m);c^*_{\mathbf{S}}]\partial_x\varphi^*(y_m))^2 \\ &+ \sum_{m=1}^{N_{sb}} |u_\theta(x_m,0)-u(x_m,0)|^2 + \sum_{m=1}^{N_{tb}} |u^*(z_m)-u(z_m)|^2 .\end{aligned} \tag{4.25}$$

4.4.2 Random Reinitialization of the Test function Parameters

(Approximate) solutions of min-max problems are significantly harder to reach, when compared to standard minimization (or maximization) problems, as they correspond to saddle points of the underlying loss function. One essential ingredient for improving the numerical stability of the algorithm is the random reinitialization of the trainable parameters η, corresponding to the test function neural network in (4.14). This can be performed with frequency r_f. This *reset frequency* can be suitably chosen as any other model hyperparameters through ensemble training. On account of this random reinitialization of the test function parameters η, the algorithm 2 can be readily modified to yield algorithm 3, that is used in practice.

4.4.3 Averages of retrainings

The final *wPINN* approximation to the solution of the scalar conservation law (4.1) at any given input (x,t), denoted as $u_{av}(x,t)$, is defined as the average over retrainings,

$$u_{av}(x,t) = \frac{1}{n_\theta}\sum_{i}^{n_\theta} u^*_i(x,t), \tag{4.26}$$

where $u_i^*(x,t)$ denotes the predictions of the underlying *wPINN* at (x,t), trained via algorithm 3, with initial parameters θ_i, η_i. This averaging is performed to yield more robust predictions as well as to provide an estimate of the underlying uncertainty in predictions, due to the random initializations of the neural network parameters during training.

Algorithm 3: *Weak PINN* training with random reset of the test function parameters

```
Result: $\theta^*_{\mathbf{S}}, \eta^*_{\mathbf{S}}, c^*_{\mathbf{S}}$
Initialize the networks $u_\theta, \varphi_\eta : D \times [0,T] \to \mathbb{R}$ and $C$;
for $i = 1, ..., e$ do
    if $i \ \% \ (r_f N) = 0$ then
        Randomly initialize $\eta$;
    end
    for $k = 1, ..., K_{max}$ do
        Compute $J_{max,C}(\theta,\eta) = \max_{c_i \in C} J_{max}(\theta,\eta,c_i)$;
        Update $\eta \leftarrow \eta + \tau_\eta \nabla J_{max,C}(\theta,\eta)$;
    end
    for $k = 1, ..., K_{min}$ do
        Compute $J_{max,C}(\theta,\eta) = \max_{c_i \in C} J_{max}(\theta,\eta,c_i)$;
        Update $\theta \leftarrow \theta - \tau_\theta \nabla(\lambda J_{max,C} + J_u)(\theta,\eta)$;
    end
end
```

4.5 Numerical Experiments

In this section, we present numerical experiments to illustrate the performance of *wPINNs*. To this end, we consider the scalar conservation law (4.1) in the domain $D = [-1,1]$, with the flux function $f(u) = \frac{1}{2}u^2$. Note that this amounts to considering the well-known inviscid Burgers' equation. We evaluate the performance of *wPINNs*, implemented through algorithm 3, by computing the (relative) total error at a final time T,

$$\mathcal{E}_r^T(\theta^*_{\mathbf{S}}) = \frac{\int_D |u^*(x,T) - u(x,T)|\, dx}{\int_D |u(x,T)|\, dx}, \tag{4.27}$$

where u^* is the prediction of the *wPINN* algorithm 3. We also compute the space-time relative error,

$$\mathcal{E}_r(\theta^*_{\mathbf{S}}) = \frac{\int_{D\times[0,T]} |u^*(x,t) - u(x,t)|\, dxdt}{\int_{D\times[0,T]} |u(x,t)|\, dxdt}, \tag{4.28}$$

to assess the performance of *wPINNs* over the entire evolution of the entropy solution. We remark that the integrals in the above error expressions can be readily approximated with Monte Carlo quadratures. We consider the following numerical experiments,

4.5.1 Standing and Moving Shock

As a first numerical example, we consider the Burgers' equation in $[-1,1] \times [0,0.5]$ with initial conditions:

$$u_0(x) = \begin{cases} 1 & x \leq 0 \\ -1 & x > 0 \end{cases}, \quad u_0(x) = \begin{cases} 1 & x \leq 0 \\ 0 & x > 0 \end{cases} \tag{4.29}$$

which result into a standing shock located at $x = 0$ and a shock moving with speed 0.5, respectively:

$$u(x,t) = \begin{cases} 1 & x \leq 0 \\ -1 & x > 0 \end{cases}, \quad u(x,t) = \begin{cases} 1 & x \leq \frac{t}{2} \\ 0 & x > \frac{t}{2} \end{cases} \tag{4.30}$$

We perform an ensemble training, as outlined in the previous section, to find the best set of hyperparameters among those mentioned in Tables 4.1 and 4.3. On the other hand, we fix $\bar{d}_\theta = 20$, $\bar{d}_\eta = 10$, $N_{min} = 1$, $\lambda = 10$ $e = 5000$, $\tau_\theta = 0.01$ and $\tau_\eta = 0.015$, $n_\theta = 10$ and the tanh activation function as the activation function σ_θ for the neural network approximating the solution of (4.1).

L_θ	L_η	σ_η	N_{max}	r_f
4,6	2,4	*sin,tanh*	6, 8	0.001, 0.005, 0.025, 0.05

Table 4.1: Hyperparameter configurations and number of retrainings employed in the ensemble training of *wPINN* for moving shock.

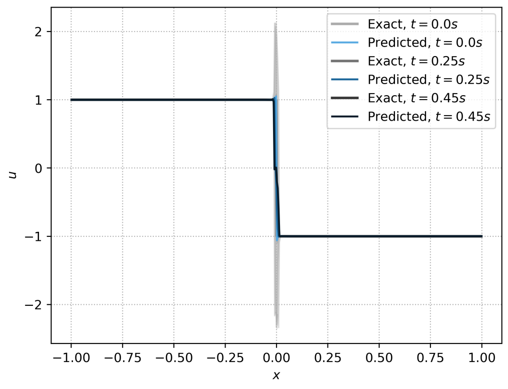
L_θ	L_η	σ_η	N_{max}	r_f
4,6	2,4	*sin,tanh*	6, 8	0.025, 0.05, 0.25

Table 4.2: Hyperparameter configurations and number of retrainings employed in the ensemble training of *wPINN* for standing shock and rarefaction wave.

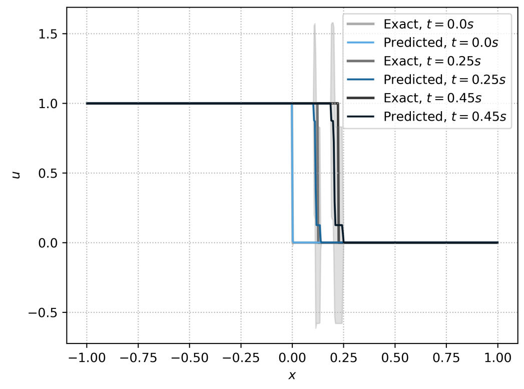
λ	N_{min}	N_{max}	r_f
0.1, 1, 10	1, 2	6, 8	0.025, 0.05, 0.25

Table 4.3: Hyperparameter configurations and number of retrainings employed in the ensemble training of *wPINN* for initial sine wave.

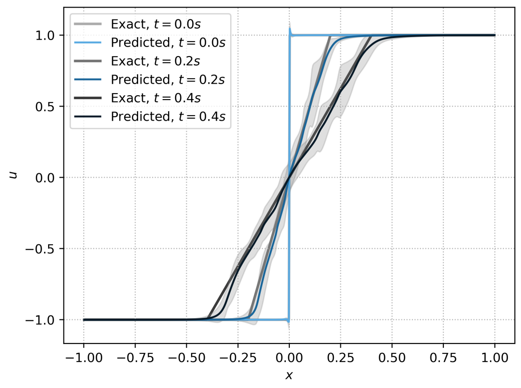
With this setting, the *wPINNs* algorithm 3 is run and its average prediction (as described in Section 4.4.3) is plotted in Figures 4.1a and 4.1b, respectively, where we also compare the predictions with the exact solutions (4.30) at different times. From these figures, we observe that the *wPINNs* average prediction accurately approximates both the standing as well as the moving shock. There is some variance in the predictions of multiple retrainings. This is completely expected as a highly non-convex min-max optimization problem is being approximated and it is possible to be trapped at local saddle points. Nevertheless, the quantitative predictions of the relative total errors $\mathcal{E}_r(\theta^*_{\mathbf{S}})$ and $\mathcal{E}^T_r(\theta^*_{\mathbf{S}})$, presented in Table 4.4, are very accurate, with even errors for the whole time-history of evolution, being below 2%. Finally, in Figure 4.2, we also plot the *wPINN* prediction that corresponds to the hyperparameter configuration that leads to smallest overall error among all tested hyperparameter configurations. We term it as the *best hyperparameter configuration.* This *best* prediction is extremely accurate, with the largest error below 0.2%. However, in practice, one does not have access to exact (or reference) solutions and needs to choose hyperparameter configurations that correspond to the smallest values of the loss function.



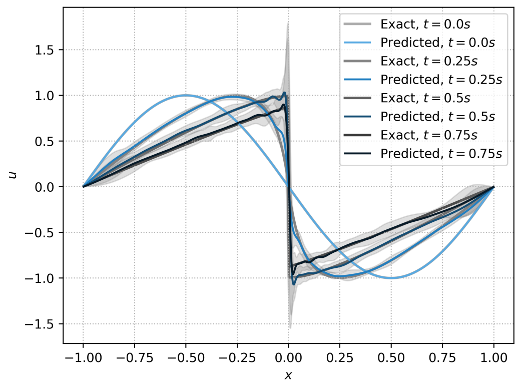
(a) Standing Shock Solution , $\mathcal{E}_r^T(\theta^*_{\mathbf{S}}) = 0.01$



(b) Moving Shock Solution, $\mathcal{E}_r^T(\theta^*_{\mathbf{S}}) = 0.019$



(c) Rarefaction Wave, $\mathcal{E}_r^T(\theta^*_{\mathbf{S}}) = 0.022$



(d) Initial Sine Wave, $\mathcal{E}_r^T(\theta^*_{\mathbf{S}}) = 0.057$

Figure 4.1: Exact solutions and average predictions obtained with *wPINN* for the Burgers' equation. Retraining average and standard deviation are plotted.

4.5.2 Rarefaction Wave

We further test the performance of *wPINNs* by considering the Burgers' equation with initial data,

$$u_0(x) = \begin{cases} -1 & x \leq 0 \\ 1 & x > 0 \end{cases} \tag{4.31}$$

The exact solution, given by,

$$u(x,t) = \begin{cases} -1 & x \leq -t \\ \frac{x}{t} & -t < x \leq t \\ 1 & x > t \end{cases} \tag{4.32}$$

corresponds to a *rarefaction wave*. We observe that this initial datum is often used to illustrate the

	N_{int}	N_{tb}	N_{sb}	$\mathcal{E}_r$	$\mathcal{E}_r^T$
Standing Shock	16384	4096	4096	0.005	0.01
Moving Shock	16384	4096	4096	0.011	0.019
Rarefaction Wave	16384	4096	4096	0.013	0.022
Initial Sine Wave	16384	4096	4096	0.03	0.057

Table 4.4: Number of training samples, total error (at final time) and total error over time, obtained with *wPINNs* (average) predictions in the numerical experiments for the Burgers' equation

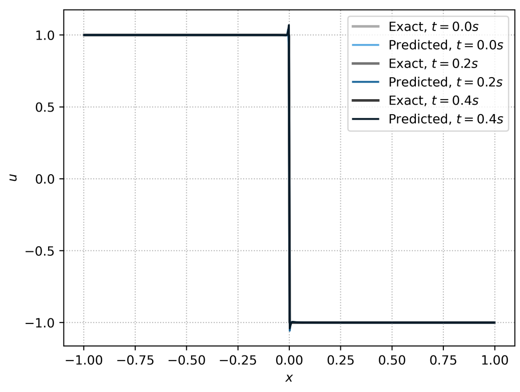
multiplicity of weak solutions of hyperbolic conservation laws as the *standing shock*, corresponding to the initial datum is clearly a weak solution but does not satisfy the entropy conditions. To illustrate the rationale behind considering *entropy residuals* (4.7) in our definitions of the loss function (3.22) in the *wPINNs* algorithm 3, we first run the same algorithm but replace the entropy residual (4.7) in Algorithm 3 with the following residuals,

$$r_{int,\theta,\eta} = \int_D \int_{[0,T]} \Big(u_{\theta,t}(x,t)\varphi_\eta(x,t) - f(u_\theta(x,t))\varphi_{\eta,x}(x,t) \Big) dtdx \tag{4.33}$$

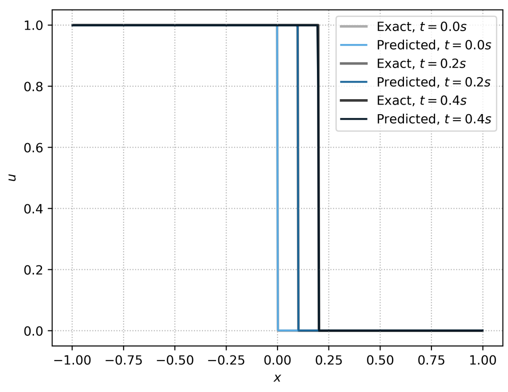
and

$$J_{int}(\theta,\eta) = \frac{\Big(\sum_{m=1}^{N_{int}} r_{int,\theta,\eta}(y_m,c) \Big)^2}{\sum_{m=1}^{N_{int}} \varphi_\eta(y_m)^2 + \partial_x \varphi_\eta(y_m)^2}, \tag{4.34}$$

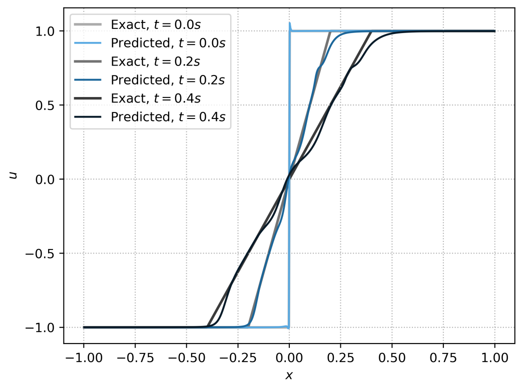
that correspond to the standard weak formulation (see definition 4.1.1) of the scalar conservation law. The resulting predictions are plotted in figure 4.3 and show that the resulting *wPINN* only approximated the *non-entropic* standing shock solution corresponding to the initial datum. Thus, a naive weak formulation of PINNs does not suffice in accurate approximations of scalar conservation laws. On the other hand, the *wPINN* average predictions of algorithm 3, with the entropy residual (4.7), provide accurate approximation of the rarefaction wave entropy solution, as shown in Figure 4.1c and Table 4.4. In particular, the error at the final time T is approximately 2% on average, whereas the error corresponding to the best hyperparameter configuration (see Figure 4.2) is only slightly lower (1.9%).



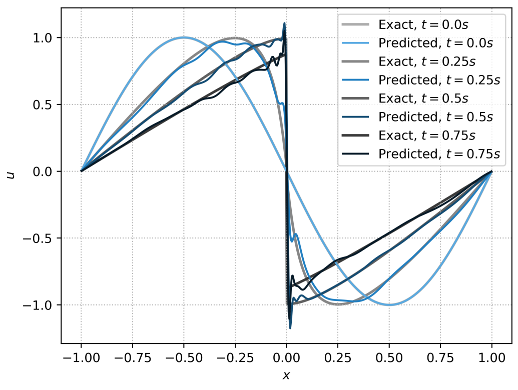
(a) Standing Shock Solution , $\mathcal{E}_r^T(\theta_{\mathbf{S}}^*) = 0.0004$



(b) Moving Shock Solution, $\mathcal{E}_r^T(\theta_{\mathbf{S}}^*) = 0.002$



(c) Rarefaction Wave, $\mathcal{E}_r^T(\theta_{\mathbf{S}}^*) = 0.019$



(d) Initial Sine Wave, $\mathcal{E}_r^T(\theta_{\mathbf{S}}^*) = 0.047$

Figure 4.2: Exact solutions and best predictions obtained with *wPINN* for the Burgers' equation.

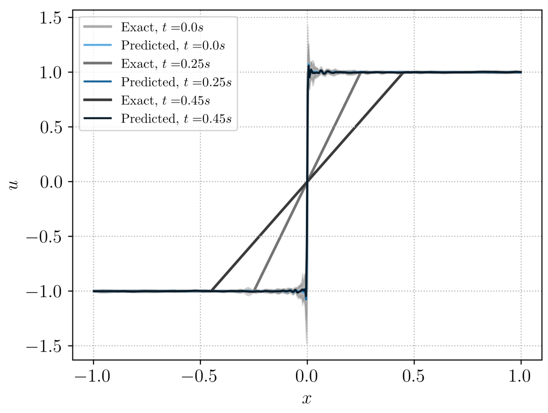


Figure 4.3: Exact rarefaction wave solution and prediction obtained with weak PINNs, without entropy conditions incorporated into the weak residual.

4.5.3 Sine Wave Initial Datum

As a final numerical experiment, we consider the Burgers' equation with the initial data,

$$u_0(x) = -\sin(\pi x)$$

and zero Dirichlet boundary conditions in the spatio-temporal domain $[-1,1] \times [0,1]$. The exact solution (approximated in Figure 4.1d with a high-resolution finite volume scheme), shows a complex evolution with both steepening as well as expansions of the sine wave that eventually form into a shock wave that separates two rarefactions. We run the *wPINNs* algorithm 3, with low-discrepancy Sobol points [72], instead of random collocation points. An ensemble training procedure, based on the hyperparameters presented in Table 4.3. The remaining parameters are set as follows: $L_\theta = 4$, $L_\eta = 2$, $\bar{d}_\theta = 20$, $\bar{d}_\eta = 10$, $\tau_\eta = 0.015$, $\tau_\theta = 0.01$ and sin activation function for both the networks. The networks are trained for $e = 75000$ epochs and parameters reinitialized $n_\theta = 15$ times, on account of the more complex underlying solution.

The (average) predictions with the *wPINNs* algorithm 3 are depicted in Figure 4.1d and show that this complicated underlying solution is approximated accurately with *wPINNs*, although there are very small spurious oscillations in the approximation. These might be further eliminated by adding additional regularization terms, such as the BV-norm into the loss function (4.14). Nevertheless, as shown in Table 4.4, the error over the entire time period is approximately 3% (8% in the L^2-norm), whereas the error at final time is understandably higher. This should also be contrasted with the relative error of approximately 24% (in the in the L^2-norm), obtained for this particular test case with conventional PINNs, as observed in [73] (Figure 6 (d)). Moreover, the error $\mathcal{E}_r^T$ is even smaller and the approximation significantly more accurate, with the best performing hyperparameter configuration shown in Figure 4.2.

5 Physics Informed Neural Networks for the Inverse Problem of PDEs

This chapter presents a comprehensive methodology for tackling inverse problems for PDEs, including the parameter identification and data assimilation problem outlined in Section 2.3, utilizing PINNs. Additionally, we provide a rigorous estimation of PINNs' generalization error in solving data assimilation problems across a broad spectrum of linear partial differential equations.

5.1 PINNs for the Parameter Identification Problem

Consider the abstract partial differential equation with boundary conditions

$$\begin{aligned} \mathcal{D}(u, a) &= s, \qquad x \in D_T \\ \mathcal{B}\left(\mathcal{T}u\right) &= f_b, \qquad x \in \partial D_T \end{aligned} \tag{5.1}$$

with $u \in \mathcal{U}$. Assume that the equation parameter $a \in \mathcal{A}$ is not known. Furthermore, we assume that we have access to (noiseless) measurements of the underlying solution u in D'_T i.e,

$$\mathcal{L}(u) = u', \tag{5.2}$$

with $\mathcal{L} : \mathcal{U}(D_T) \to \mathcal{Z}(D'_T)$ being the *observation operator* defined in Section 2.3. The parameter identification problem for 5.1 consists in finding $a \in \mathcal{A}$ such that (5.2) and (5.1) are satisfied.

In order to solve the parameter identification problem with physic informed neural networks, we approximate both $u \in \mathcal{U}$ and $a \in \mathcal{A}$ with neural networks (2.24) with parameters $\theta, \eta \in \Theta$,

$$u_\theta(x) \approx u(x), \quad a_\eta(x) \approx a(x), \quad x \in D_T. \tag{5.3}$$

We then define the following residuals:

$$\begin{aligned} &\textit{PDE (or Interior) residual} && r_{int}[u_\theta, a_\eta] := \mathcal{D}(u_\theta, a_\eta) - s, \quad x \in D_T \\ &\textit{Boundary residual} && r_{bc}[u_\theta] := \mathcal{B}(\mathcal{T}u_\theta) - f_b, \quad x \in \partial D_T \\ &\textit{Data residual} && r_d[u_\theta] := \mathcal{L}(u_\theta) - u', \quad x \in D'_T \end{aligned} \tag{5.4}$$

and corresponding losses,

$$J_{int}(\theta, \eta) := \|r_{int}[u_\theta, a_\eta]\|^{p_s}_{\mathcal{S}(D_T)}, \quad J_{bc}(\theta) := \|r_{bc}[u_\theta]\|^{p_b}_{\mathcal{B}(\partial D_T)}, \qquad J_d(\theta) := \|r_d[u_\theta]\|^{p_z}_{\mathcal{Z}(D'_T)} \tag{5.5}$$

which we approximate with the quadrature rule on the quadrature points $\mathcal{S}_{int} = \{x_n\}$ with $x_n \in D_T$ for all $1 \le n \le N$, and $\mathcal{S}_{bc} = \{y_m, f_b(y_m)\}$, $y_m \in \partial D_T$, for all $1 \le m \le M$ and $\mathcal{S}_d = \{z_\ell, u'(z_\ell)\}$, $z_\ell \in \partial D_T$ for all $1 \le \ell \le L$:

$$J_{int}(\theta, \eta) := \sum_{n=1}^{N} w_n |r_{int}[u_\theta, a_\eta](x_m)|^{p_s}, \quad J_{bc}(\theta) := \sum_{m=1}^{M} w_m |r_{bc}[u_\theta](y_m)|^{p_b}, \quad J_d(\theta) := \sum_{\ell=1}^{L} w_\ell |r_d[u_\theta](z_\ell)|^{p_z} \tag{5.6}$$

Finally, we solve the following optimization problem

$$\theta^*, \eta^* = \arg\min_{\theta,\eta\in\Theta} J(\theta), \tag{5.7}$$

with

$$J(\theta,\eta) = J_{int}(\theta,\eta) + \lambda J_u(\theta), \quad J_u(\theta) = J_{bc}(\theta) + J_d(\theta) \tag{5.8}$$

The algorithm for training PINNs can then be described as follows.

Algorithm 4: Finding a physics informed neural network to approximate the solution of the parameter identification problem (5.1), (5.2)

Input

$\mathcal{D}$	differential operator on the underlying domain D_T
s	input source term
f_b	boundary conditions
u'	observable
$\mathcal{S}_{int}$	training set for approximating the PDE residual norm
$\mathcal{S}_{bc}$	training set for approximating the boundary residual norm
$\mathcal{S}_d$	training set for approximating the data residual norm
$\overline{\theta}$	initial value of the tunable parameters for u_θ
$\overline{\eta}$	initial value of the tunable parameters for a_η
λ	balancing parameter

Output

u_{θ^*}	PINNs approximating u in the parameter identification problem (5.1), (5.2)
a_{η^*}	PINNs approximating a in the parameter identification problem (5.1), (5.2)

Step 1: For the initial value of the weight vectors $\overline{\theta}, \overline{\eta} \in \Theta$, evaluate the neural network $u_{\overline{\theta}}$, $a_{\overline{\eta}}$ (2.24), the PDE, boundary, and data residuals (5.4), the loss function (5.8) and its gradients.
Step 2: Run the optimization algorithm till an approximate local minimum θ^*, η^* of (5.8) is reached. The maps $u^* = u_{\theta^*}$ and $a^* = a_{\eta^*}$ are the desired PINNs for approximating the solution (u, a) of the parameter identification problem (5.1), (5.2).

5.2 PINNs for the Data Assimilation Problem

Similar to the parameter identification problem, consider the partial differential equation

$$\mathcal{D}(u) = s, \qquad x \in D_T \tag{5.9}$$

with $u \in \mathcal{U}$ and assume that we have access to (noiseless) measurements of the underlying solution u in a *sub-domain* $D'_T \subset D_T$ i.e,

$$\mathcal{L}(u) = u', \quad \forall x \in D'_T, \tag{5.10}$$

In the data assimilation problem the generic boundary conditions (which might include initial conditions) f_b are not known. Thus, in solving the inverse problem (5.9), (5.10), one determines the function $u \in \mathcal{U}$ and consequently the boundary conditions $\mathcal{B}(\mathcal{T}u)$ from the data (5.10) such that (5.9) is satisfied.

We assume that solutions to the data assimilation problem (5.9), (5.10), satisfy the following estimate,

Definition 5.2.1 (Conditional Stability). *Let $\hat{\mathcal{U}} \subset \mathcal{U} \subset L^{p_u}(D_T)$ be Banach spaces. For any $u, v \in \hat{\mathcal{U}}$, the differential operator $\mathcal{D}$ and restriction operator $\mathcal{L}$ satisfy,*

$$\|u - v\|_{L^{p_u}(E)} \leq C_{pd}\left(\|u\|_{\hat{\mathcal{U}}}, \|v\|_{\hat{\mathcal{U}}}\right)\left(\|\mathcal{D}(u) - \mathcal{D}(v)\|_{\mathcal{S}}^{\tau_p} + \|\mathcal{L}(u) - \mathcal{L}(v)\|_{\mathcal{Z}}^{\tau_d}\right), \tag{5.11}$$

for some $0 < \tau_p, \tau_d \leq 1$ and for any subset $D'_T \subset E \subset D_T$.

This bound (5.11) is termed as *conditional stability estimate* as it presupposes that the underlying solutions have sufficiently regularity as $\hat{\mathcal{U}} \subset \mathcal{U} \subset L^{p_u}(D_T)$.

The stability condition can be also be replaced by the following weaker one,

Definition 5.2.2 (Weak Conditional Stability). *Let $\hat{\mathcal{U}} \subset \mathcal{U} \subset L^{p_u}(D_T)$ be Banach spaces. For any $u, v \in \mathcal{U}(D_T)$, the differential operator $\mathcal{D}$ and restriction operator $\mathcal{L}$ satisfy,*

$$\|u - v\|_{L^{p_u}(E)} \leq C_{pd}\left(\|u\|_{\hat{\mathcal{U}}}, \|v\|_{\hat{\mathcal{U}}}\right)\omega\left(\|\mathcal{D}(u) - \mathcal{D}(v)\|_{\mathcal{S}} + \|\mathcal{L}(u) - \mathcal{L}(v)\|_{\mathcal{Z}}\right), \tag{5.12}$$

for any subset $D'_T \subset E \subset D_T$, with $\omega : \mathbb{R} \to \mathbb{R}_+$ being a modulus of continuity.

Observe that the stability estimate (5.12) is a specialization of the general stability estimate (2.12), where the roles of the PDE and data are made explicit.

In order to approximate a solution to the data assimilation problem with PINNs, we proceed as before and choose a feed-forward neural network (2.24) as possible solution:

$$u_\theta(x) \approx u(x), \quad x \in D_T \tag{5.13}$$

Next, we define the following residual

$$\begin{array}{lll} \textit{PDE (or Interior) residual} & r_{int}[u_\theta] := \mathcal{D}(u_\theta) - s, & x \in D_T \\ \textit{Data residual} & r_d[u_\theta] := \mathcal{L}(u_\theta) - u', & x \in D'_T \end{array} \tag{5.14}$$

and loss

$$J(\theta) = J_{int}(\theta) + \lambda J_d(\theta) \tag{5.15}$$

with

$$J_{int}(\theta) := \|r_{int}[u_\theta]\|_{\mathcal{S}(D_T)}^{p_s}, \qquad J_d(\theta) := \|r_d[u_\theta]\|_{\mathcal{Z}(D'_T)}^{p_u} \tag{5.16}$$

The strategy of PINNs, following Sections 3.1 and 5.1, is to minimize the resulting loss over the admissible set of tuning parameters $\theta \in \Theta$ i.e

$$\theta^* = \arg\min_{\theta \in \Theta} J(\theta), \tag{5.17}$$

in order to find an approximate solution to the data assimilation problem. The algorithm for training PINNs is summarized in Algorithm 5.

Remark 5.2.3. *One of the most notable features of PINNs is their ability to approximate solutions of* inverse problems*, with the same accuracy and computational cost as that of forward problems for PDEs. Observe, in fact, that approximating both the forward and inverse problems for PDEs with PINNs simplifies to (1) approximate the solution of the problem with a neural network* (2.24) *and (2) simultaneously minimize the losses J_{int} and J_u, with J_u including any available type of data, for instance boundary data only (as in the forward problem), observation data (as in data assimilation problem) or boundary and observation data (as in the parameter identification problem).*

```
Algorithm 5: Finding a physics informed neural network to approximate the solution of the data
assimilation problem (5.9), (5.10)

Input
    $\mathcal{D}$          differential operator on the underlying domain $D_T$
    $s$          input source term
    $u'$         observable
    $\mathcal{S}_{int}$       training set for approximating the PDE residual norm
    $\mathcal{S}_d$         training set for approximating the data residual norm
    $\overline{\theta}$          initial value of the tunable parameters for $u_\theta$
    $\lambda$          balancing parameter
Output
    $u_{\theta^*}$       PINN approximating $u$ in the parameter identification problem (5.9), (5.10)

Step 1: For the initial value of the weight vector $\overline{\theta} \in \Theta$, evaluate the neural network $u_{\overline{\theta}}$ (2.24), the
PDE and data residuals (5.14), the loss function (5.15) and its gradients.
Step 2: Run the optimization algorithm until an approximate local minimum $\theta^*$ of (5.15) is reached.
The map $u^* = u_{\theta^*}$ is the desired PINN for approximating the solution $u$ of the data assimilation
problem (5.9), (5.10).
```

5.2.1 An abstract estimate on the generalization error

In this section, we will estimate the error due to the PINN in approximating the solution u of the inverse problem for PDE (5.9), (5.10).

We focus on the so-called *generalization error* of the PINN

$$\mathcal{E}_G(E) = \mathcal{E}_G(E; \theta^*, \mathcal{S}_{int}, \mathcal{S}_d) := \|u - u^*\|_{\mathcal{U}(E)}, \tag{5.18}$$

with $D'_T \subset E \subset D_T$. As written out above, the generalization error clearly depends on the training sets $\mathcal{S}_{int}, \mathcal{S}_d$, as well as on the parameters θ^* of the PINN, generated by algorithm 5. However, we shall suppress this explicit dependence for notational convenience.

We estimate the generalization error (5.18) in terms of the so-called *training error* $\mathcal{E}_T = \mathcal{E}_T(\theta^*, \mathcal{S}_{int}, \mathcal{S}_d)$ defined by,

$$\mathcal{E}_T = \left((\mathcal{E}_T^d)^{p_z} + (\mathcal{E}_T^{int})^{p_s}\right)^{\frac{2}{p_s+p_z}}, \quad \mathcal{E}_T^d := \left(\sum_{\ell=1}^{L} w_\ell^d |r_d[u_\theta](z_\ell)|^{p_z}\right)^{\frac{1}{p_z}}, \quad \mathcal{E}_T^{int} := \left(\sum_{n=1}^{N} w_i^{int} |r_{int}[u_\theta](y_i)|^{p_s}\right)^{\frac{1}{p_s}}. \tag{5.19}$$

Note that, after the training has concluded, the training error $\mathcal{E}_T$ can be readily computed from the loss functions (5.16).

The bound on generalization error in terms of training error is given by the following estimate,

Theorem 5.2.4. *Let $u \in \hat{\mathcal{U}} \subset \mathcal{U} \subset L^{p_u}(D_T)$ be the solution of the inverse problem associated with PDE* (5.9) *and data* (5.10)*. Assume that the stability hypothesis* (5.11) *holds for any $D'_T \subset E \subset D_T$. Let $u^* \in \hat{\mathcal{U}} \subset \mathcal{U}$ be a PINN generated by minimizing* (5.15)*, based on the training sets $\mathcal{S}_{int}$ and $\mathcal{S}_d$ (of quadrature points corresponding to the quadrature rule* (2.19)*). Further assume that the residuals $r_{int}[u_{\theta^*}]$*

and $r_{int}[u_{\theta^*}]$, defined in (5.14)*, be such that $r_{int}[u_{\theta^*}] \in \mathcal{S}(D_T)$ and $r_d[u_{\theta^*}] \in \mathcal{Z}(D'_T)$ and the quadrature errors satisfy* (2.20)*. Then the following estimate on the generalization error* (5.18) *holds,*

$$\mathcal{E}_G \leq C_{pd}\left((\mathcal{E}_T^{int})^{\tau_p} + (\mathcal{E}_T^{d})^{\tau_d} + (C_{quad}^{int})^{\frac{\tau_p}{p_s}} N^{-\frac{\alpha\tau_p}{p_s}} + (C_{quad}^{d})^{\frac{\tau_d}{p_z}} L^{-\frac{\alpha\tau_d}{p_z}} \right), \tag{5.20}$$

with constants $C_{pd} = C_{pd}\left(\|u\|_{\hat{\mathcal{U}}}, \|u^*\|_{\hat{\mathcal{U}}}\right)$, $C_{quad}^{int} = C_{quad}^{int}\left(\|r_{int}[u_{\theta^*}]\|_{\mathcal{S}(D_T)}\right)$, and $C_{quad}^{d} = C_{quad}^{d}\left(\|r_d[u_{\theta^*}]\|_{\mathcal{Z}(D'_T)}\right)$ stem from (5.11) *and* (2.20)*, respectively.*

Proof. For notational simplicity, we denote $r_{int} = r_{int}[u_{\theta^*}]$, the residual (3.2), corresponding to the trained neural network u^*. Similarly, $r_d = r_d[u_{\theta^*}]$ is the data residual (5.14), corresponding to u^*.

As u solves the PDE (5.9) and r_{int} is defined by (5.14), we easily see that,

$$r_{int} = \mathcal{D}(u^*) - \mathcal{D}(u). \tag{5.21}$$

Similarly, u satisfies the data relation (5.10) and by definition (5.14), we have,

$$r_d = \mathcal{L}(u^*) - \mathcal{L}(u). \tag{5.22}$$

By our assumptions, PINN $u^* \in \hat{\mathcal{U}}$, hence, we can directly apply the *conditional stability estimate* (5.11) and use (5.21), (5.22) to obtain,

$$\begin{aligned} \mathcal{E}_G(E) &= \|u - u^*\|_{L^{p_x}(E)} \quad \text{(by (5.18))}, \\ &\leq C_{pd}\left(\|\mathcal{D}(u^*) - \mathcal{D}(u)\|_{\mathcal{S}(D_T)}^{\tau_p} + \|\mathcal{L}(u) - \mathcal{L}(u^*)\|_{\mathcal{Z}}^{\tau_d} \right) \quad \text{(by (5.11))}, \\ &\leq C_{pd}\left(\|r_{int}\|_{\mathcal{S}(D_T)}^{\tau_p} + \|r_d\|_{\mathcal{Z}}^{\tau_d} \right) \quad \text{(by (5.21), (5.22))}. \end{aligned} \tag{5.23}$$

By the fact that $\mathcal{S}(D_T) \subset L^{p_s}(D_T)$, the definition of the training error (5.19) and the quadrature rule (2.19), we see that,

$$\|r_{int}\|_{\mathcal{S}(D_T)}^{p_s} \approx \sum_{n=1}^{N} w_n^{int} |r_{int}(x_n)|^{p_s} = (\mathcal{E}_T^{int})^{p_s}.$$

Hence, the training error component $\mathcal{E}_T^{int}$ is a quadrature for the residual (3.2) and the resulting quadrature error, given by (2.20) translates to,

$$\|r_{int}\|_{\mathcal{S}(D_T)}^{p_s} \leq (\mathcal{E}_T^{int})^{p_s} + C_{quad}^{int} N^{-\alpha},$$

and as $\tau_p \leq 1$

$$\|r_{int}\|_{\mathcal{S}(D_T)}^{\tau_p} \leq (\mathcal{E}_T^{int})^{\tau_p} + (C_{quad}^{int})^{\frac{\tau_p}{p_s}} N^{-\frac{\alpha\tau_p}{p_s}}. \tag{5.24}$$

Similarly as $\mathcal{Z}(D'_T) \subset L^{p_z}(D'_T)$, the definition of the training error (5.19) and the quadrature rule (2.19), we have that,

$$\|r_d\|_{\mathcal{Z}}^{p_z} \approx \sum_{\ell=1}^{L} w_\ell^{d} |r_d(z_\ell)|^{p_z} = (\mathcal{E}_T^{d})^{p_z}.$$

Hence, the training error component $\mathcal{E}_T^{d}$ is a quadrature for the residual (5.14) and the resulting quadrature error, given by (2.20) leads to,

$$\|r_d\|_{\mathcal{Z}}^{p_z} \leq (\mathcal{E}_T^{d})^{p_z} + C_{quad}^{d} L^{-\alpha},$$

and as $\tau_d \leq 1$

$$\|r_d\|_{\mathcal{Z}}^{\tau_d} \leq (\mathcal{E}_T^d)^{\tau_d} + (C_{quad}^d)^{\frac{\tau_d}{p_z}} L^{-\frac{\alpha\tau_d}{p_z}}. \tag{5.25}$$

Substituting (5.24) and (5.25) into (5.23) yields the desired bound (5.20). □

We term a PINN, generated by the algorithm 5 to be *well-trained* if the following condition hold,

$$\max\left\{(\mathcal{E}_T^{int})^{\tau_p}, (\mathcal{E}_T^d)^{\tau_d}\right\} \leq (C_{quad}^{int})^{\frac{\tau_p}{p_s}} N^{-\frac{\alpha\tau_p}{p_s}} + (C_{quad}^d)^{\frac{\tau_d}{p_z}} L^{-\frac{\alpha\tau_d}{p_z}}. \tag{5.26}$$

Thus, a well-trained PINN is one for which the training errors are smaller than the so-called *generalization gap* (given by the righ-hand-side of (5.26)).

The following remarks about Theorem 5.2.4 are in order.

Remark 5.2.5. *The estimate* (5.20) *indicates mechanisms that underpin possible efficient approximation of solutions of inverse (unique continuation, data assimilation) problems by PINNs as it breaks down the sources of error into the following parts,*

- *The PINN has to be well-trained i.e, the training error $\mathcal{E}_T$ has to be sufficiently small. Note that we have no a priori control on the training error but can compute it* a posteriori.
- *The class of approximating PINNs has to be sufficiently regular such that the residuals in* (5.14) *can be approximated to high accuracy by the quadrature rules* (2.19)*. This regularity of PINNs can be enforced by choosing smooth activation functions such as the sigmoid and hyperbolic tangent in* (2.24)*.*
- *Finally, the whole estimate* (5.20) *relies on the conditional stability estimate* (5.11) *for the inverse problem* (5.9)*,* (5.10)*. Thus, the generalization error estimate leverages conditional stability for inverse problems of PDEs into efficient approximation by PINNs. This is very similar to the conclusion outlined for the forward problem of PDE (Section 3.2).*

Remark 5.2.6. *We note that the estimate* (5.20) *on the error due to PINNs contains the constants $C_{pd}, C_{quad}^{int}, C_{quad}^d$. These constants depend on the underlying solution but also on the trained neural network u^* and on the residuals. For any given number of training samples N, L, these constants are bounded as the underlying neural networks and residuals are smooth functions on bounded domains. However, as $N, L \to \infty$, these constants might blow-up. As long as these constants blow up at rates that are slower wrt N, L than the decay terms in* (5.20)*, one can expect that the overall bound* (5.20) *still decays to zero as $N, L \to \infty$. In practice, one has a finite number of training samples and the bounds on the constants can be verified a posteriori from the computed residuals and trained neural networks.*

Remark 5.2.7. *The estimate* (5.20) *was based on deterministic quadrature rules* (2.19)*. Following section 2.4.1 of [73], it can be extended, in a straightforward manner, to the case of randomly chosen training points that stem from Monte Carlo quadrature.*

Remark 5.2.8. *We have considered a noiseless measurement in the data term* (2.11) *on the observation domain. However, the bound* (5.20) *can be readily extended to cover the noisy case in the following manner. We assume that the measurements for the inverse problem are given by,*

$$\mathcal{L}(u) = u' + \eta, \tag{5.27}$$

for a noise term $\eta \in \mathcal{Z}$. Then, a straightforward modification of the argument in the proof of Theorem 5.2.4 yields the bound,

$$\mathcal{E}_G \leq C_{pd}\left((\mathcal{E}_T^{int})^{\tau_p} + (\mathcal{E}_T^d)^{\tau_d} + \|\eta\|_{L^{p_z}(D'_T)}^{\tau_d} + (C_{quad}^{int})^{\frac{\tau_p}{p_s}} N^{-\frac{\alpha\tau_p}{p_s}} + (C_{quad}^{d})^{\frac{\tau_d}{p_z}} L^{-\frac{\alpha\tau_d}{p_z}}\right), \tag{5.28}$$

with constants already defined in (5.20)*. Thus, as long as the noise term is small in magnitude, the PINN will still efficiently approximate solution of the inverse problem. However, for N, N_d sufficiently small, the noise term will dominate* (5.28) *and one has to use a Bayesian framework to approximate solutions of the inverse problem in this regime.*

5.3 Poisson's equation

As a first example for the abstract inverse problem (5.9), (5.10), we consider the Poisson's equation as a model problem for linear elliptic PDEs.

5.3.1 The underlying inverse problem

Let $D \subset \mathbb{R}^d$ be an open, bounded, simply connected set with smooth boundary ∂D[1]. We consider the Poisson's equation on this domain,

$$-\Delta u = f, \quad \forall x \in D, \tag{5.29}$$

with Δ denoting the Laplace operator and $f \in L^2(D)$ being a source term. We will assume that $u \in H^1(D)$ will satisfy the Poisson's equation (5.29) in the following weak sense,

$$\int_D \nabla u \cdot \nabla v dx = \int_D f v dx, \tag{5.30}$$

for all test functions $v \in H_0^1(D)$. Note that (5.30) follows as a consequence of multiplying the test function $v \in H_0^1(D)$ and integrating by parts. Moreover, the PDE (5.29) is not well-posed as u is not necessarily in $H_0^1(D)$.

The unique continuation (data assimilation) inverse problem in this case is given by,

$$u|_{D'} = g, \quad \text{for some } D' \subset D, \tag{5.31}$$

with $g \in H^1(D')$ and the observation domain D' being open, simply connected and with a smooth boundary $\partial D'$.

Formally, in order for the unique continuation problem (5.29), (5.31) to have a solution, it is necessary that g satisfies the Poisson's equation (5.29) in D'. Hence, the unique continuation problem is formally equivalent to,

$$\begin{aligned} -\Delta u &= f, \quad \forall x \in D \setminus D', \\ u &= g, \quad \forall x \in \partial D', \\ \nabla u \cdot n &= \nabla g \cdot n, \quad \forall x \in \partial D'. \end{aligned} \tag{5.32}$$

The problem (5.32) is termed as the *Elliptic Cauchy problem* and was already studied by Hadamard as an example of ill-posed problems for PDEs.

Well-posedness results for the inverse problem (5.29), (5.31) are classical, see [74] for a detailed survey. Here, we follow the slightly simplified presentation due to [75] and state the following result,

[1]Further geometric conditions on the boundary might be necessary for obtaining bounds on the quadrature error

Theorem 5.3.1. *[[75], Theorem 7.1]: Let $f \in L^2(D)$ and let $u \in H^1(D)$ such that* (5.30) *holds for all test functions $v \in H^1_0(D)$. Let $g \in H^1(D')$ such that* (5.31) *holds, then for every open simply connected set $E \subset D$ such that dist$(E, \partial D) > 0$, there holds,*

$$\|u\|_{H^1(E)} \leq C\left(\|u\|_{H^1(D)}\right)\omega\left(\|f\|_{L^2(D)} + \|g\|_{L^2(D')}\right). \tag{5.33}$$

Here, $C(R) = CR^{1-\tau}$ and $\omega(R) = R^\tau$, for some absolute constant C and exponent $\tau \in (0,1)$, depending on the set E.

Moreover, we have the global stability estimate,

$$\|u\|_{H^1(D)} \leq C\left(\|u\|_{H^1(D)}\right)\omega\left(\|f\|_{L^2(D)} + \|g\|_{L^2(D')}\right), \tag{5.34}$$

with the same $C(R)$ as in (5.33)*, but with modulus of continuity given by $\omega(R) = |\log(R)|^{-\tau}$, with $\tau \in (0,1)$.*

The theorem, as presented in [75] (Theorem 7.1) has slightly better estimates than (5.33), (5.34). In particular, on the righ-hand-side, the norm on the source term is the H^{-1} norm. However, the current version of these estimates suffices for our purposes here. Moreover, detailed derivation of the constants is given in Theorem 4.4 of [74] and the proof of (5.33), (5.34) is based on the *three-balls* inequality.

Remark 5.3.2. *Relating the above formulation of the unique continuation inverse problem* (5.29)*,* (5.31) *to the abstract formalism presented in section 5.2.1, we see that with $\mathcal{U} \subset L^2(D), \mathcal{S} \subset L^2(D), \mathcal{Z} \subset L^2(D')$, the differential operator $\mathcal{D} = -\Delta$, interpreted weakly, and the observable $\mathcal{L} = ID$, it is straightforward to bound the abstract conditional stability estimate* (5.12) *in the concrete forms* (5.33) *or* (5.34)*. Thus, this problem falls squarely in the framework considered in section 5.2.1 and can be approximated by PINNs.*

5.3.2 PINNs

Training sets

As the training set $\mathcal{S}_{int}$ in algorithm 5, we take a set of quadrature points $y_i \in D$, for $1 \leq i \leq N_{int}$, corresponding to the quadrature rule (2.19). These can be quadrature points for a grid based (composite) Gauss quadrature rule or low-discrepancy sequences such as Sobol points. Similarly, the training set $\mathcal{S}_d = \{z_j, g(z_j)\}$ for $z_j \in D'$, with $1 \leq j \leq N_d$, are quadrature points, corresponding to the quadrature rule (2.19).

Residuals

We will require that for parameters $\theta \in \Theta$, the neural networks $u_\theta \in C^k(D)$, for $k \geq 2$. This can be enforced by choosing a sufficiently regular activation function in (2.24). We define the following residuals that are needed in algorithm 5. The PDE residual (3.2) is given by,

$$r_{int}[u_\theta] = -\Delta u_\theta - f, \quad \forall x \in D, \tag{5.35}$$

and the data residual on the observation domain is given by,

$$r_d[u_\theta] = u_\theta - g, \quad \forall x \in D'. \tag{5.36}$$

Loss functions

In algorithm 5 for approximating the inverse problem (5.29), (5.31), we will need the following loss function,

$$J(\theta) = \lambda \sum_{j=1}^{N_d} w_j^d |r_d[u_\theta](z_j)|^2 + \sum_{i=1}^{N_{int}} w_i^{int} |r_{int}[u_\theta](y_i)|^2, \tag{5.37}$$

with hyperparamter λ, residuals defined in (5.35), (5.36), training points defined above and weights w_i^{int}, w_j^d, corresponding to quadrature rule (2.19).

5.3.3 Estimates on the generalization error

We consider any $E \subset D$, with E open, simply connected and such that $dist(E, \partial D) > 0$ and define the generalization error with respect to the PINN $u^* = u_{\theta^*}$, generated by the algorithm 5 with training sets, residuals and loss functions described above, as

$$\mathcal{E}_G(E) = \|u - u^*\|_{H^1(E)}. \tag{5.38}$$

As in theorem 3.2.1, this generalization error will be estimated in terms of the following training errors,

$$\mathcal{E}_T^d = \left(\sum_{j=1}^{N_d} w_j^d |r_d[u_{\theta^*}](z_j)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_T^{int} = \left(\sum_{i=1}^{N_{int}} w_i^{int} |r_{int}[u_{\theta^*}](y_i)|^2 \right)^{\frac{1}{2}}. \tag{5.39}$$

Note that the training errors $\mathcal{E}_T^{int}$ and $\mathcal{E}_T^d$, can be readily computed from the loss functions (3.6), (5.37), a posteriori. We have the following estimate on the generalization error in terms of the training error,

Lemma 5.3.3. *For $f \in C^{k-2}(D)$ and $g \in C^k(D')$, with continuous extensions of the functions and derivatives up to the boundaries of the underlying sets and with $k \geq 2$, Let $u \in H^1(D)$ be the solution of the inverse problem corresponding to the Poisson's equation* (5.29) *i.e, it satisfies* (5.30) *for all test functions $v \in H_0^1(D)$ and satisfies the data* (5.31). *Let $u^* = u_{\theta^*} \in C^k(D)$ be a PINN generated by the algorithm 5, with loss functions* (3.6), (5.37). *Then, the generalization error* (5.38) *for any any $E \subset D$, with E open, simply connected and such that $dist(E, \partial D) > 0$ is bounded by,*

$$\|u - u^*\|_{H^1(E)} \leq C \left(\|u\|_{H^1(D)}^{1-\tau} + \|u^*\|_{H^1(D)}^{1-\tau} \right) \left((\mathcal{E}_T^{int})^\tau + (\mathcal{E}_T^d)^\tau + (C_{quad}^{int})^{\frac{\tau}{2}} N_{int}^{-\frac{\alpha\tau}{2}} + (C_{quad}^d)^{\frac{\tau}{2}} N_d^{-\frac{\alpha\tau}{2}} \right), \tag{5.40}$$

for some $\tau \in (0,1)$ and constant C depending on E and with constants $C_{quad}^{int} = C_{quad}^{int}\left(\|r_{int}[u_{\theta^*}]\|_{C^{k-2}(D)}\right)$ and $C_{quad}^d = C_{quad}^d\left(\|r_d[u_{\theta^*}]\|_{C^k(D')}\right)$, given by the quadrature error bound (2.20).

Moreover, we also have the global error bound,

$$\|u - u^*\|_{H^1(D)} \leq C \left(\|u\|_{H^1(D)}^{1-\tau} + \|u^*\|_{H^1(D)}^{1-\tau} \right) \left| \log \left(\mathcal{E}_T^{int} + \mathcal{E}_T^d + (C_{quad}^{int})^{\frac{1}{2}} N_{int}^{-\frac{\alpha}{2}} + (C_{quad}^d)^{\frac{1}{2}} N_d^{-\frac{\alpha}{2}} \right) \right|^{-\tau}. \tag{5.41}$$

Proof. For notational simplicity, we denote $r_{int} = r_{int}[u_{\theta^*}]$ and $r_d = r_d[u_{\theta^*}]$.

Define $\hat{u} = u^* - u \in H^1(D)$, by linearity of the differential operator in (5.29) and the data observable in (5.31) and by definitions (5.35), (5.36), we see that $\hat{u}$ satisfies,

$$\begin{aligned} -\Delta\hat{u} &= r_{int}, \quad \forall x \in D, \\ \hat{u} &= r_d, \quad \forall x \in D', \end{aligned} \tag{5.42}$$

with Poisson equation being satisfied in the following weak sense,

$$\int\limits_D \nabla\hat{u} \cdot \nabla v dx = \int\limits_D r_{int} v dx, \tag{5.43}$$

for all test functions $v \in H_0^1(D)$. Hence, we can directly apply the conditional stability estimate (5.33) to obtain for some $\tau \in (0,1)$,

$$\begin{aligned} \|\hat{u}\|_{H^1(E)} &\leq C(\|\hat{u}\|_{H^1(D)}^{1-\tau}) \left(\|r_d\|_{L^2(D')} + \|r_{int}\|_{L^2(D)}\right)^\tau \\ &\leq C\left(\|u\|_{H^1(D)}^{1-\tau} + \|u^*\|_{H^1(D)}^{1-\tau}\right) \left(\|r_d\|_{L^2(D')} + \|r_{int}\|_{L^2(D)}\right)^\tau . \end{aligned} \tag{5.44}$$

Recognizing that the training errors $(\mathcal{E}_T^d)^2$ and $(\mathcal{E}_T^{int})^2$ are the quadratures for $\|r_d\|_{L^2(D')}^2$ and $\|r_{int}\|_{L^2(D)}^2$ with respect to the quadrature rule (2.19), respectively and using bound (2.20) yields,

$$\begin{aligned} \|r_d\|_{L^2(D')} &\leq \mathcal{E}_T^d + (C_{quad}^d)^{\frac{1}{2}} N_d^{\frac{\alpha}{2}}, \\ \|r_{int}\|_{L^2(D)} &\leq \mathcal{E}_T^{int} + (C_{quad}^{int})^{\frac{1}{2}} N_{int}^{\frac{\alpha}{2}}, \end{aligned} \tag{5.45}$$

with constants $C_{quad}^{int} = C_{quad}^{int}\left(\|r_{int}\|_{C^{k-2}(D)}\right)$ and $C_{quad}^d = C_{quad}^d\left(\|r_d\|_{C^k(D')}\right)$.

It is straightforward to obtain the desired inequality (5.40) by substituting (5.45) into (5.44).

The bound (5.41) can be obtained, completely analogously, by replacing (5.33) in (5.44) with (5.34). □

5.4 Heat Equation

As a model inverse problem for linear parabolic PDEs, we will consider the data assimilation problem for the heat equation.

5.4.1 The underlying inverse problem

With $D \subset \mathbb{R}^d$ being an open, bounded, simply connected set with smooth boundary ∂D, we consider the heat equation with zero Dirichlet boundary conditions,

$$\begin{aligned} u_t - \Delta u &= f, \quad \forall x \in D, t \in (0,T), \\ u = h &= 0, \quad \forall x \in \partial D, t \in (0,T), \end{aligned} \tag{5.46}$$

for some $T \in \mathbb{R}_+$, with Δ denoting the *spatial* Laplace operator and $f \in L^2(D_T)$ with $D_T = D \times (0,T)$ being the source term. We also assume zero Dirichlet boundary conditions for simplicity.

We will assume that $u \in H^1(((0,T); H^{-1}(D)) \cap L^2((0,T); H^1(D))$ will solve the heat equation (5.46) in a weak sense.

The heat equation (5.46) would have been well-posed if the initial conditions i.e $u_0 = u(x,0)$ had been specified. In fact, the aim of the *data assimilation* problem is to infer the initial conditions and the entire solution field at later times from some measurements of the solution in time. To model this, we consider the following *observables*:

$$u = g, \quad \forall (x,t) \in D' \times (0,T), \tag{5.47}$$

for some open, simply connected observation domain $D' \subset D$ and for $g \in L^2(D'_T)$ with $D'_T = D' \times (0,T)$.

Thus solving the data assimilation problem (5.46), (5.47), amounts to finding the solution u of the heat equation (5.46) in the whole space-time domain D_T, given data on the observation sub-domain D'_T. The theory for this data assimilation inverse problem for the heat equation is classical and several well-posedness and stability results are available. Our subsequent error estimates for PINNs rely on the following classical result of Imanuvilov [76], based on the well-known Carleman estimates,

Theorem 5.4.1. *[76]: Let $u \in H^1((0,T);H^{-1}(D)) \cap L^2((0,T);H^1(D))$ solve the heat equation* (5.46) *with source term $f \in L^2(D_T)$, then for every $0 \le \bar{T} < T$, there holds,*

$$\|u\|_{C([\bar{T},T];L^2(D))} + \|u\|_{L^2((0,T);H^1(D))} \le C\left(\|u\|_{L^2(D'_T)} + \|f\|_{L^2(D_T)} + \|u\|_{L^2(\partial D\times(0,T))}\right), \tag{5.48}$$

for some constant $C > 0$.

Remark 5.4.2. *Relating the above formulation of the data assimilation inverse problem* (5.46)*,* (5.47) *to the abstract formalism presented in section 5.2.1, we readily see that $\mathcal{U} \subset L^2(D_T), \mathcal{S} \subset L^2(D_T), \mathcal{Z} \subset L^2(D'_T)$, the differential operator is $\mathcal{D} = \partial_t - \Delta$, interpreted weakly, and the observable is $\mathcal{L} = ID$. Moreover, the abstract conditional stability estimate* (5.12) *takes the concrete form* (5.48)*. Thus, this problem falls squarely in the framework considered in section 5.2.1 and can be approximated by PINNs.*

5.4.2 PINNs

Training sets

The training set $\mathcal{S}_d = \{z_j, g(z_j)\}$ for $z_j = (x_j,t_j) \in D'_T$, with $1 \le j \le N_d$, are quadrature points, corresponding to the quadrature rule (2.19). Given the fact that we also consider boundary conditions in (5.46), we need to slightly modify the training set on which the PDE residual (3.2) is going to be collocated. We take *interior* training set $\mathcal{S}_{int}$ as set of quadrature points $y_i = (x_i,t_i) \in D_T$, for $1 \le i \le N_{int}$, corresponding to the quadrature rule (2.19). These can be quadrature points for a grid based (composite) Gauss quadrature rule or low-discrepancy sequences such as Sobol points. We also need to introduce *spatial boundary* training set $\mathcal{S}_{sb} = \{\bar{y}_i, h(\bar{y}_i)\}$ for $1 \le i \le N_{sb}$, with $\bar{y}_i = (\bar{x}_i,t_i)$, and $t_i \in (0,T)$, $\bar{x}_i \in \partial D$, for each i. These points can be quadrature points corresponding to a *boundary quadrature rule* i.e for any function $h : \partial D \times (0,T) \to \mathbb{R}$, we approximate

$$\int_0^T \int_{\partial D} h(x,t)\,d\sigma(x)\,dt \approx \sum_{i=1}^{N_{sb}} w_i^{sb} h(\bar{x}_i,t_i), \tag{5.49}$$

and we also assume that this boundary quadrature rule satisfies an error estimate of the form,

$$\left| \int_0^T \int_{\partial D} h(x,t)\,d\sigma(x)\,dt - \sum_{i=1}^{N_{sb}} w_i^{sb} h(\bar{x}_i,t_i) \right| \le C_{bd} N_{sb}^{-\alpha_{sb}}, \tag{5.50}$$

with constant $C_{bd} = C\left(\|h\|_{C^\ell(\partial D\times(0,T))}\right)$.

Residuals

We will require that for parameters $\theta \in \Theta$, the neural networks $(x,t) \mapsto u_\theta(x,t) \in C^k(\overline{D}_T)$, for $k \geq 2$. We define the following residuals that are needed in algorithm 5. The PDE residual (3.2) is given by,

$$r_{int}[u_\theta] = \partial_t u_\theta - \Delta u_\theta - f, \quad \forall (x,t) \in D_T. \tag{5.51}$$

We need the following residual to account for the boundary data in (5.46),

$$r_{sb}[u_\theta] = u_\theta|_{\partial D \times (0,T)}, \quad \forall (x,t) \in \partial D \times (0,T) \tag{5.52}$$

and the data residual is given by,

$$r_d[u_\theta] = u_\theta - g, \quad \forall (x,t) \in D'_T. \tag{5.53}$$

Loss functions

In algorithm 5 for approximating the inverse problem (5.46), (5.47), we will need the following loss function,

$$J(\theta) = \sum_{j=1}^{N_d} w_j^d |r_d[u_\theta](z_j)|^2 + \sum_{i=1}^{N_{sb}} w_i^{sb} |r_{sb}[u_\theta](\bar{y}_i)|^2 + \lambda \sum_{i=1}^{N_{int}} w_i^{int} |r_{int}[u_\theta](y_i)|^2, \tag{5.54}$$

with hyperparamter λ, residuals defined in (5.51), (5.52), (5.53), training points defined above and weights w_i^{int}, w_i^{sb}, w_j^d, corresponding to quadrature rules (2.19), (5.49), respectively.

5.4.3 Estimates on the generalization error

For any $0 \leq \bar{T} < T$, we define the generalization error for the PINN $u^* = u_{\theta^*}$, generated by algorithm 5 to approximate the solution u to the data assimilation, inverse problem (5.46), (5.47) as,

$$\mathcal{E}_G(\bar{T}) = \|u - u^*\|_{C([\bar{T},T];L^2(D))} + \|u - u^*\|_{L^2((0,T);H^1(D))}. \tag{5.55}$$

We estimate this generalization error in terms of the following training errors,

$$\mathcal{E}_T^d = \left(\sum_{j=1}^{N_d} w_j^d |r_d[u_{\theta^*}](z_j)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_T^{int} = \left(\sum_{i=1}^{N_{int}} w_i^{int} |r_{int}[u_{\theta^*}](y_i)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_T^{sb} = \left(\sum_{i=1}^{N_{sb}} w_i^{sb} |r_{sb}[u_\theta](\bar{y}_i)|^2 \right)^{\frac{1}{2}}. \tag{5.56}$$

Note that the training errors $\mathcal{E}_T^{int}, \mathcal{E}_T^{sb}$ and $\mathcal{E}_T^d$, can be readily computed from the loss functions (3.6), (5.54), a posteriori. We have the following estimate on the generalization error in terms of the training error,

Lemma 5.4.3. *For $f \in C^{k-2}(D_T)$ and $g \in C^k(D'_T)$, with continuous extensions of the functions and derivatives upto the boundaries of the underlying sets and with $k \geq 2$, let $u \in H^1((0,T);H^{-1}(D)) \cap L^2((0,T);H^1(D))$ be the solution of the inverse problem corresponding to the heat equation* (5.46) *and satisfies the data* (5.47)*. Let $u^* = u_{\theta^*} \in C^k(D_T)$ be a PINN generated by the algorithm 5, with loss functions* (3.6), (5.54)*. Then, the generalization error* (5.55) *for any $0 \leq \bar{T} < T$ is bounded by,*

$$\mathcal{E}_G(\bar{T}) \leq C \left(\mathcal{E}_T^d + \mathcal{E}_T^{int} + \mathcal{E}_T^{sb} + C_q^{\frac{1}{2}} N_{int}^{-\frac{\alpha}{2}} + C_{bd}^{\frac{1}{2}} N_{sb}^{-\frac{\alpha_{sb}}{2}} + C_{qd}^{\frac{1}{2}} N_d^{-\frac{\alpha_d}{2}} \right), \tag{5.57}$$

for some constant C depending on $\bar{T}, u, u^*$ and with constants $C_q = C_q\left(\|r_{int}[u_{\theta^*}]\|_{C^{k-2}(D_T)}\right)$, $C_{bd} = C_{bd}\left(\|r_{sb}[u_{\theta^*}]\|_{C^k(\partial D\times(0,T))}\right)$ and $C_{qd} = C_{qd}\left(\|r_d[u_{\theta^*}]\|_{C^k(D'_T)}\right)$, given by the quadrature error bounds (2.20)*,* (5.50)*.*

Proof. For notational simplicity, we denote $r_{int} = r_{int}[u_{\theta^*}], r_{sb} = r_{sb}[u_{\theta^*}]$ and $r_d = r_d[u_{\theta^*}]$.

Define $\hat{u} = u^* - u \in H^1((0,T);H^{-1}(D)) \cap L^2((0,T);H^1(D))$, by linearity of the differential operator in (5.46) and the data observable in (5.47) and by definitions (5.51), (5.52) and (5.53), we see that $\hat{u}$ satisfies,

$$\begin{aligned} \hat{u}_t - \Delta\hat{u} &= r_{int}, \quad \forall x,t \in D_T, \\ \hat{u} &= r_{sb}, \quad \forall x \in \partial D,\ t \in (0,T), \\ \hat{u} &= r_d, \quad \forall x,t \in D'_T. \end{aligned} \tag{5.58}$$

Hence, we can directly apply the conditional stability estimate (5.48) to obtain,

$$\|\hat{u}\|_{C([\bar{T},T];L^2(D))} \le C\left(\|r_d\|_{L^2(D')_T} + \|r_{int}\|_{L^2(D_T)} + \|r_{sb}\|_{L^2(\partial D\times(0,T))}\right), \tag{5.59}$$

with the constant C from (5.48).

Recognizing that the training errors $(\mathcal{E}_T^d)^2, (\mathcal{E}_T^{sb})^2$ and $(\mathcal{E}_T^{int})^2$ are the quadrature approximations for $\|r_d\|^2_{L^2(D')_T}$, $\|r_{sb}\|_{L^2(\partial D\times(0,T))}$ and $\|r_{int}\|^2_{L^2(D_T)}$ with respect to the quadrature rules (5.49) and (2.19), respectively and using bounds (5.50) and (2.20) yields the inequality (5.45). Substituting it into (5.59) leads to the desired bound (5.57) in a straightforward manner.

□

5.5 The Wave equation

Next, we will consider the wave equation as a model problem for linear hyperbolic PDEs.

5.5.1 The underlying inverse problem

With $D \subset \mathbb{R}^d$ being an open, bounded, simply connected set with smooth boundary ∂D, we consider the wave equation with zero Dirichlet boundary conditions,

$$\begin{aligned} u_{tt} - \Delta u &= f, \quad \forall (x,t) \in D \times (0,T), \\ u &= h \equiv 0, \quad \forall (x,t) \in \partial D \times (0,T), \end{aligned} \tag{5.60}$$

for some $T \in \mathbb{R}_+$, with Δ denoting the *spatial* Laplace operator and $f \in L^2(D_T)$ with $D_T = D \times (0,T)$ being the source term.

The forward problem for the wave equation is only well-posed when we know the initial conditions,

$$\begin{aligned} u(x,0) &= u_0(x), \quad \forall x \in D, \\ u_t(x,0) &= u_1(x), \quad \forall x \in D, \end{aligned} \tag{5.61}$$

for some $u_0 \in L^2(D)$ and $u_1 \in H^{-1}(D)$. However, in many problems of interest, the initial data $u_{0,1}$ are not known and have to be inferred from measurements of the form,

$$u(x,t) = g, \quad (x,t) \in D' \times (0,T), \tag{5.62}$$

for some subset $D' \subset \bar{D}$. We also denote the observation domain as $D'_T = D' \times (0,T)$.

The resulting *data assimilation* inverse problem consists of finding the initial data u_0, u_1 and consequently the entire solution field u of the wave equation (5.60), given data f, g. A slight variant of this problem stems from photoacoustic tomography (PAT) [77] and it also arises in control theory, in the form of so-called Luenberger observers [78].

The data assimilation problem for the wave equation (5.60), (5.62) has received considerable amount of attention in the mathematical literature and can be solved as long as the following *geometric control condition*, introduced in the seminal paper [79], is satified,

Definition 5.5.1. Geometric Control Condition*(GCC) (see [79, 80]): The domain $D'_T \subset \bar{D}_T$, is said to satisfy the geometric control condition (gcc) in D_T if every compressed generalized bicharacteristic $(x(s), t(s), \tau(s), \xi(s))$ intersects the set D'_T for some $s \in \mathbb{R}$*

We refer the reader to [79, 80] for the rather technical definition of generalized bicharacteristics. It roughly states that all light rays in D_T must intersect D'_T, taking reflections at the boundary into account. An even stronger, sufficient condition that implies the GCC is the so-called Γ-condition that [81] roughly requires that the final time T and the set $\partial D' \cap \partial D$ are relatively large.

Under the GCC, one can follow [82] and prove the following well-posedness result for the data assimilation problem for the wave equation (5.60),(5.62),

Theorem 5.5.2. *[[82],Theorem 2.2]: Let $D_T = D \times (0,T)$, such that $D \subset \mathbb{R}^d$ is a domain with smooth boundary ∂D. Let $D'_T = D' \times (0,T)$, with $D' \subset \bar{D}$, satisfy the geometric control condition. If $u \in L^2(D_T)$ be such that $u(\cdot,0) \in L^2(D)$, $\partial_t u(\cdot,0) \in H^{-1}(D)$, $u|_{\partial D \times (0,T)} \in L^2(\partial D \times (0,T))$, then the following estimate holds,*

$$\sup_{t \in [0,T]} \left(\|u(\cdot,t)\|_{L^2(D)} + \|\partial_t u(\cdot,t)\|_{H^{-1}(D)} \right) \leq \kappa \left(\|u\|_{L^2(D'_T)} + \|f\|_{L^2(D_T)} + \|u\|_{L^2(\partial D \times (0,T))} \right), \tag{5.63}$$

with observability constant κ that depends on the underlying domain geometry and final time T. The proof of this theorem relies heavily on the so-called observability estimates of [79], which are derived by using micro-local propagation of singularities for the wave equation. Alternative proofs use Carleman estimates [83].

Again, we can recast the data assimilation inverse problem for the wave equation in the abstract formulation of section 5.2.1, with (5.63) playing the role of the conditional stability estimate (5.12). Thus, this inverse problem is amenable to efficient approximation by PINNs.

5.5.2 PINNs

We specify the algorithm 5 to generate a PINN for approximating the data assimilation inverse problem (5.60), (5.62), in the following steps,

Training sets

We consider exactly the same training sets as for the heat equationn i.e, $\mathcal{S}_d = \{z_j, g(z_j)\}$ for $z_j = (x_j, t_j) \in D'_T$, with $1 \leq j \leq N_d$, are quadrature points, corresponding to the quadrature rule (2.19), $\mathcal{S}_{int}$ as set of quadrature points $y_i = (x_i, t_i) \in D_T$, for $1 \leq i \leq N_{int}$, corresponding to the quadrature rule (2.19) and *spatial boundary* training set $\mathcal{S}_{sb} = \{\bar{y}_i, h(\bar{y}_i)\}$ for $1 \leq i \leq N_{sb}$, with $\bar{y}_i = (\bar{x}_i, t_i)$, and $t_i \in (0,T)$, $\bar{x}_i \in \partial D$, for each i. These points can be quadrature points corresponding to the *boundary quadrature rule* (5.49).

Residuals

We will require that for parameters $\theta \in \Theta$, the neural networks $(x,t) \mapsto u_\theta(x,t) \in C^k(\overline{D_T})$, for $k \geq 2$. We define the following residuals that are needed in algorithm 5. The PDE residual (3.2) is given by,

$$r_{int}[u_\theta] = \partial_{tt} u_\theta - \Delta u_\theta - f, \quad \forall (x,t) \in D_T. \tag{5.64}$$

We need the following residual to account for the boundary data in (5.60),

$$r_{sb}[u_\theta] = u_\theta|_{\partial D \times (0,T)}, \quad \forall (x,t) \in \partial D \times (0,T). \tag{5.65}$$

The data residual is given by,

$$r_d[u_\theta] = u_\theta - g, \quad \forall (x,t) \in D'_T. \tag{5.66}$$

Loss functions

In algorithm 5 for approximating the inverse problem (5.60), (5.62), we will need the following loss function (which is identically the same as the loss function (5.54) for the heat equation),

$$J(\theta) = \sum_{j=1}^{N_d} w_j^d |r_d[u_\theta](z_j)|^2 + \sum_{i=1}^{N_{sb}} w_i^{sb} |r_{sb}[u_\theta](\bar{y}_i)|^2 + \lambda \sum_{i=1}^{N_{int}} w_i^{int} |r_{int}[u_\theta](y_i)|^2, \tag{5.67}$$

with hyperparamter λ, residuals defined in (5.66), (5.64) and (5.65), training points defined above and weights w_j^d, w_i^{int} and w_i^{sb}, corresponding to quadrature rules (2.19) and (5.49), respectively.

5.5.3 Estimates on the generalization error

Denoting $u^* = u_{\theta^*}$ as the PINN, generated by algorithm 5 to approximate the data assimilation problem (5.60), (5.62), for the wave equation, we define the following generalization error,

$$\mathcal{E}_G := \|u - u^*\|_{C^1([0,T];H^{-1}(D)) \cap C([0,T];L^2(D))} = \sup_{t \in [0,T]} \left(\|u(\cdot,t) - u^*(\cdot,t)\|_{L^2(D)} + \|\partial_t u(\cdot,t) - u^*(\cdot,t)\|_{H^{-1}(D)} \right). \tag{5.68}$$

As in theorem 3.2.1 and the previous examples, we will bound the generalization error in terms of the following training errors,

$$\mathcal{E}_T^d = \left(\sum_{j=1}^{N_d} w_j^d |r_d[u_{\theta^*}](z_j)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_T^{int} = \left(\sum_{i=1}^{N_{int}} w_i^{int} |r_{int}[u_{\theta^*}](y_i)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_T^{sb} = \left(\sum_{i=1}^{N_{sb}} w_i^{sb} |r_{sb}[u_{\theta^*}](\bar{y}_i)|^2 \right)^{\frac{1}{2}}. \tag{5.69}$$

The training errors $\mathcal{E}_T^{int}, \mathcal{E}_T^{sb}$ and $\mathcal{E}_T^{d}$, can be readily computed from the loss functions (3.6), (5.67), a posteriori. We have the following estimate on the generalization error in terms of the training error,

Lemma 5.5.3. *Let the domain D'_T satisfy the geometric control condition in D_T. For $f \in C^{k-2}(D_T)$ and $g \in C^k(D'_T)$, with $k \geq 2$, let $u \in C([0,T)]; L^2(D)) \cap C^1([0,T]; H^{-1}(D))$ be the solution of the data assimilation corresponding to the wave equation* (5.60) *and satisfies the data* (5.62)*. Let $u^* = u_{\theta^*} \in C^k(D_T)$ be a PINN generated by the algorithm 5, with loss functions* (3.6), (5.67)*. Then, the generalization error* (5.68) *is bounded by,*

$$\mathcal{E}_G \leq \kappa \left(\mathcal{E}_T^d + \mathcal{E}_T^{int} + \mathcal{E}_T^{sb} + C_q^{\frac{1}{2}} N_{int}^{-\frac{\alpha}{2}} + C_{bd}^{\frac{1}{2}} N_{sb}^{-\frac{\alpha_{sb}}{2}} + C_{qd}^{\frac{1}{2}} N_d^{-\frac{\alpha_d}{2}} \right), \tag{5.70}$$

for observability constant κ depending on the underlying geometry, T, u, u^* and with constants $C_q = C_q\left(\|r_{int}[u_{\theta^*}]\|_{C^{k-2}(D_T)}\right), C_{bd} = C_{bd}\left(\|r_{sb}[u_{\theta^*}]\|_{C^k(\partial D\times(0,T))}\right)$ and $C_{qd} = C_{qd}\left(\|r_d[u_{\theta^*}]\|_{C^k(D'_T)}\right)$, given by the quadrature error bounds (2.20), (5.50), *respectively.*

Proof. For notational simplicity, we denote $r_{int} = r_{int}[u_{\theta^*}], r_{sb} = r_{sb}[u_{\theta^*}]$ and $r_d = r_d[u_{\theta^*}]$.

Define $\hat{u} = u^* - u \in C^1([0,T]; H^{-1}(D)) \cap C([0,T]; L^2(D))$, by linearity of the differential operator and boundary conditions in (5.60) and the data observable in (5.62) and by definitions (5.64),(5.65),(5.53), we see that $\hat{u}$ satisfies,

$$\begin{aligned} \hat{u}_{tt} - \Delta \hat{u} &= r_{int}, \quad \forall (x,t) \in D_T, \\ \hat{u}|_{\partial D \times (0,T)} &= r_{sb}, \\ \hat{u} &= r_d, \quad \forall (x,t) \in D'_T. \end{aligned} \tag{5.71}$$

Therefore, we can apply the observability estimate (5.63) to obtain,

$$\sup_{t\in[0,T]} \left(||\hat{u}(\cdot,t)||_{L^2(D)} + \|\partial_t \hat{u}(\cdot,t)\|_{H^{-1}(D)} \right) \leq \kappa \left(\|r_d\|_{L^2(D'_T)} + \|r_{int}\|_{L^2(D_T)} + \|r_{sb}\|_{L^2(\partial D\times(0,T))} \right), \tag{5.72}$$

with the observability constant κ from (5.63).

Realizing that the training errors $(\mathcal{E}_T^d)^2$, $(\mathcal{E}_T^{int})^2$ and $(\mathcal{E}_T^{sb})^2$ are the quadrature approximations for $\|r_d\|^2_{L^2(D')_T}$, $\|r_{int}\|^2_{L^2(D_T)}$and $\|r_{sb}\|^2_{L^2(\partial D\times(0,T))}$ with respect to the quadrature rules (2.19) and (5.49), respectively and using bounds (2.20) and (5.50) and substituting the result into (5.72) yields the desired bound (5.70). □

5.6 The Stokes equation

The effectiveness of PINNs in approximating inverse problems was brilliantly showcased in the recent paper [17], where the authors proposed PINNs for the data assimilation problem with the Navier-Stokes equation. As a first step towards rigorously analyzing this, we will focus on the much simpler model of the stationary Stokes equation below.

5.6.1 The underlying inverse problem

Let $D \subset \mathbb{R}^d$ be an open, bounded, simply connected set with smooth boundary. We consider the Stokes' equations as a model of stationary, highly viscous fluid:

$$\begin{aligned} \Delta u + \nabla p &= f, \quad \forall x \in D, \\ \nabla \cdot u &= f_d, \quad \forall x \in D. \end{aligned} \tag{5.73}$$

Here, $u : D \mapsto \mathbb{R}^d$ is the velocity field, $p : D \mapsto \mathbb{R}$ is the pressure and $f : D \mapsto \mathbb{R}^d$, $f_d : D \mapsto \mathbb{R}$ are source terms.

Note that the Stokes equation (5.73) is not well-posed as we are not providing any boundary conditions. In the corresponding data assimilation problem [84] and references therein, one provides the following *data*,

$$u = u', \quad \forall x \in D', \tag{5.74}$$

for some open, simply connected set $D' \subset D$. Thus, the data assimilation inverse problem for the Stokes equation amounts to inferring the velocity field u (and the pressure p)), given f, f_d and u'. In particular, we wish to find solutions $u \in H^1(D; \mathbb{R}^d)$ and $p \in L^2_0(D)$ (i.e, square integrable functions with zero mean), such that the following holds,

$$\begin{aligned} \int_D \nabla u \cdot \nabla v dx + \int_D p \nabla \cdot v dx &= \int_D f v dx, \\ \int_D \nabla \cdot u w dx &= \int_D f_d w dx, \end{aligned} \tag{5.75}$$

for all test functions $v \in H^1_0(D; \mathbb{R}^d)$ and $w \in L^2(D)$.

The well-posedness and conditional stability estimates for the data assimilation problem for the Stokes equation (5.73), (5.74) has been extensively investigated in [85] and references therein. In particular, we have the following stability estimate,

Theorem 5.6.1. *Let $D' \subset D$ and let $B_{R_1}(x_0)$ be the largest ball, $B_{R_1}(x_0) \subset D'$. Let $u \in H^1(D; \mathbb{R}^d)$ and $p \in L^2_0(D)$ satisfy (5.75) for all test functions $v \in H^1_0(D; \mathbb{R}^d)$ and $w \in L^2(D)$. Then for any $f \in L^2(D; \mathbb{R}^d)$, $f_d \in L^2(D)$ and for any $R_2 > R_1$ such that $B_{R_2}(x_0) \subset D$, the following bound holds,*

$$\|u\|^2_{L^2(B_{R_2}(x_0))} \leq C \left(C_{f,f_d} + C_u^\tau C_{f,f_d}^{1-\tau} + \left(C_{f,f_d}^{1-\tau} C_u^{1-\tau} \right) \|u\|^{2\tau}_{L^2(D')} \right), \tag{5.76}$$

with constants defined as,

$$C_{f,f_d} := \|f\|^2_{L^2(D)} + \|f_d\|^2_{L^2(D)}, \quad C_u := \|u\|^2_{L^2(D)}. \tag{5.77}$$

Proof. Let u_1, p_1 be solutions of the following boundary-value problem for the Stokes equations,

$$\begin{aligned} \Delta u_1 + \nabla p_1 &= f, \quad \forall x \in D, \\ \nabla \cdot u_1 &= f_d, \quad \forall x \in D, \\ u_1 &\equiv 0, \quad \forall x \in \partial D. \end{aligned} \tag{5.78}$$

We know from classical theory for the Stokes' equation [86], [84](estimate 2.2) that,

$$\|u_1\|_{H^1(D)} + \|p_1\|_{L^2(D)} \leq C \left(\|f\|_{L^2(D)} + \|f_d\|_{L^2(D)} \right), \tag{5.79}$$

for some constant C that depends only on the domain D.

Define u_2, p_2 as the solutions of the following homogeneous Stokes' equations,

$$
\begin{aligned}
\Delta u_2 + \nabla p_2 &= 0, \quad \forall x \in D, \\
\nabla \cdot u_2 &= 0, \quad \forall x \in D.
\end{aligned} \tag{5.80}
$$

We know that for any $R_1 < R_2 < R_3$, such that $B_{R_2}(x_0) \subset D$, the following *three balls* inequality [85] holds for the solutions $u_2 \in H^1(D; \mathbb{R}^d)$ and $p_2 \in H^1(D)$ of the homogeneous Stokes equations (5.80),

$$
\int_{B_{R_2}(x_0)} |u_2|^2 dx \le C \left(\int_{B_{R_1}(x_0)} |u_2|^2 dx \right)^{\tau} \left(\int_{B_{R_3}(x_0)} |u_2|^2 dx \right)^{1-\tau}, \tag{5.81}
$$

with a constant C and $\tau \in (0,1)$ that depends on $\frac{R_1}{R_3}$, $\frac{R_2}{R_3}$ and d (see [85] for more on the constants).

It is easy to check that $u = u_1 + u_2$ and $p = p_1 + p_2$, satisfy the Stokes equation (5.73) in the weak sense, i.e they satisfy (5.75). By identifying all constants in the following calculation with a generic constant C, we have,

$$
\begin{aligned}
\int_{B_{R_2}(x_0)} |u|^2 dx &= \int_{B_{R_2}(x_0)} |u_1 + u_2|^2 dx \\
&\le C \left(\int_{B_{R_2}(x_0)} |u_1|^2 dx + \int_{B_{R_2}(x_0)} |u_2|^2 dx \right) \le C \left(\int_{D} |u_1|^2 dx + \int_{B_{R_2}(x_0)} |u_2|^2 dx \right) \\
&\overset{\text{(by (5.79), (5.81))}}{\le} C \left(\|f\|^2_{L^2(D)} + \|f_d\|^2_{L^2(D)} + \left(\int_{B_{R_1}(x_0)} |u_2|^2 dx \right)^{\tau} \left(\int_{B_{R_3}(x_0)} |u_2|^2 dx \right)^{1-\tau} \right) \\
&\le C \left(\|f\|^2_{L^2(D)} + \|f_d\|^2_{L^2(D)} + \left(\int_{B_{R_1}(x_0)} |u - u_1|^2 dx \right)^{\tau} \left(\int_{B_{R_3}(x_0)} |u - u_1|^2 dx \right)^{1-\tau} \right) \\
&\le C \left(\|f\|^2_{L^2(D)} + \|f_d\|^2_{L^2(D)} + \|u\|^{2\tau}_{L^2(B_{R_1}(x_0))} \|u\|^{2(1-\tau)}_{L^2(D)} \right) \\
&+ C \left(\|u\|^{2\tau}_{L^2(B_{R_1}(x_0))} \|u_1\|^{2(1-\tau)}_{L^2(D)} + \|u_1\|^{2\tau}_{L^2(D)} \|u\|^{2(1-\tau)}_{L^2(D)} + \|u_1\|^2_{L^2(D)} \right).
\end{aligned}
$$

Substituting (5.79) and identifying the constants yields the desired bound (5.76).

□

As in the previous examples, this inverse problem can be recast in terms of the abstract formalism of section 5.2.1, with the conditional stability estimate (5.76) occupying the role of (5.12) and making this inverse problem amenable to efficient approximation by PINNs.

5.6.2 PINNs

We specify the algorithm 5 to generate a PINN for approximating the inverse problem (5.73), (5.74) in the following steps,

Training sets

As the training set $\mathcal{S}_{int}$ in algorithm 5, we take a set of quadrature points $y_i \in D$, for $1 \leq i \leq N_{int}$, corresponding to the quadrature rule (2.19). These can be quadrature points for a grid based (composite) Gauss quadrature rule or low-discrepancy sequences such as Sobol points. Similarly, the training set $\mathcal{S}_d = \{z_j, g(z_j)\}$ for $z_j \in D'$, with $1 \leq j \leq N_d$, are quadrature points, corresponding to the quadrature rule (2.19).

Residuals

We will require that for parameters $\theta \in \Theta$, the neural networks $u_\theta \in C^k(D;\mathbb{R}^d)$, $p_\theta \in C^k(D;\mathbb{R})$, for $k \geq 2$. We define the following residuals that are needed in algorithm 5. The PDE residual (3.2), consists of two parts in this case (see also [73], section 5), given by,

$$r_{int}[u_\theta] = \Delta u_\theta + \nabla p_\theta - f, \quad \forall x \in D, \tag{5.82}$$

and

$$r_{div}[u_\theta] = \nabla \cdot u_\theta - u', \quad \forall x \in D, \tag{5.83}$$

The data residual is given by,

$$r_d[u_\theta] = u_\theta - u', \quad \forall x \in D'. \tag{5.84}$$

Loss functions

In algorithm 5 for approximating the inverse problem (5.73), (5.74), we will need the following loss function,

$$J(\theta) = \sum_{j=1}^{N_d} w_j^d |r_d[u_\theta](z_j)|^2 + \lambda\Bigg(\sum_{i=1}^{N_{int}} w_i^{int}|r_{int}[u_\theta](y_i)|^2 + \sum_{i=1}^{N_{int}} w_i^{int}|r_{div}[u_\theta](y_i)|^2\Bigg), \tag{5.85}$$

with hyperparamter λ, residuals defined in (5.35), (5.36), training points defined above and weights w_i^{int}, w_j^d, corresponding to quadrature rule (2.19).

5.6.3 Estimates on the generalization error

Let $B_{R_1}(x_0)$ be the largest ball inside the observation domain $D' \subset D$. We will consider balls $B_R(x_0) \in D$ such that $R > R_1$ and estimate the generalization error,

$$\mathcal{E}_G(B_R) := \|u - u^*\|_{L^2(B_R(x_0))}, \tag{5.86}$$

with $u^* = u_{\theta^*}$ is the PINN generated by algorithm 5. We will estimate this generalization error in terms of the training errors defined by,

$$\mathcal{E}_T^d = \left(\sum_{j=1}^{N_d} w_j^d |r_d[u_{\theta^*}](z_j)|^2\right)^{\frac{1}{2}}, \quad \mathcal{E}_T^{int} = \left(\sum_{i=1}^{N_{int}} w_i^{int} r_{int}[u_{\theta^*}](y_i)|^2 + \sum_{i=1}^{N_{int}} w_i^{int}|r_{div}[u_\theta](y_i)|^2\right)^{\frac{1}{2}}, \tag{5.87}$$

in the following lemma,

Lemma 5.6.2. *For $f \in C^{k-2}(D;\mathbb{R}^d), f_d \in C^{k-1}(D)$ and $u' \in C^k(D')$, with $k \geq 2$. Let $u \in H^1(D;\mathbb{R}^d)$ and $p \in H^1(D)$ be the solution of the inverse problem corresponding to the Stokes' equations* (5.73) *i.e, they satisfy* (5.75) *for all test functions $v \in H_0^1(D;\mathbb{R}^d), w \in L^2(D)$ and satisfies the data* (5.74)*. Let $u^* = u_{\theta^*} \in C^k(D;\mathbb{R}^d), p^* = p_{\theta^*} \in C^k(D)$ be a PINN generated by the algorithm 5, with loss functions* (3.6), (5.85)*. Let $B_{R_1}(x_0)$ be the largest ball inside $D' \subset D$. Then, the generalization error* (5.86) *for balls $B_R(x_0) \in D$ with $R > R_1$ is bounded by,*

$$\begin{aligned}&(\mathcal{E}_G(B_R))^2 \leq \\ &C\Big[(\mathcal{E}_T^{int})^2 + C_{\hat{u}}^{\tau}(\mathcal{E}_T^{int})^{2(1-\tau)} + C_{\hat{u}}^{1-\tau}(\mathcal{E}_T^{int})^{2(1-\tau)}(\mathcal{E}_T^{d})^{2\tau} + \\ &+ C_q N^{-\alpha} + C_{\hat{u}}^{\tau} C_q^{(1-\tau)} N^{-(1-\tau)\alpha} + C_{\hat{u}}^{1-\tau} C_q^{(1-\tau)} C_{qd}^{\tau} N^{-(1-\tau)\alpha} N_d^{-\alpha_d \tau}\Big]\end{aligned} \tag{5.88}$$

with constants $C_{\hat{u}} = \|u\|^2_{L^2(D)} + \|u^*\|^2_{L^2(D)}$, $C_q = C_q\left(res_{int}[u_{\theta^*}]\|_{C^{k-2}(D)} + \|r_{div}[u_{\theta^*}]\|_{C^{k-1}(D)}\right)$ and $C_{qd} = C_{qd}\left(\|r_d[u_{\theta^*}]\|_{C^k(D')}\right)$, given by the quadrature error bound (2.20) *and τ given in* (5.76).

Proof. For notational simplicity, we denote $r_{int} = r_{int}[u_{\theta^*}]$, $r_{div} = r_{div}[u_{\theta^*}]$ and $r_d = r_d[u_{\theta^*}]$.

Define $\hat{u} = u^* - u \in H^1(D;\mathbb{R}^d)$ and $\hat{p} = p^* - p \in H^1(D)$, by linearity of the differential operator in (5.73) and the data observable in (5.74) and by definitions (5.82), (5.83), (5.84), we see that $\hat{u}, \hat{p}$ satisfy,

$$\begin{aligned}\Delta\hat{u} + \nabla\hat{p} &= r_{int}, \quad \forall x \in D, \\ \nabla \cdot \hat{u} &= r_{div}, \quad \forall x \in D, \\ \hat{u} &= r_d, \quad \forall x \in D',\end{aligned} \tag{5.89}$$

with Stokes' equation being satisfied in the sense of (5.75).

Hence, we can directly apply the conditional stability estimate (5.76) to obtain,

$$\mathcal{E}_G^2(B_R) \leq C\left(C_{r_{int}} + C_{\hat{u}}^{\tau} C_{r_{int}}^{1-\tau} + \left(C_{r_{int}}^{1-\tau} C_{\hat{u}}^{1-\tau}\right)\|r_d\|^{2\tau}_{L^2(D')}\right), \tag{5.90}$$

with constants,

$$C_{r_{int}} = \|r_{int}\|^2_{L^2(D)} + \|r_{div}\|^2_{L^2(D)}, \quad C_{\hat{u}} = \|u\|^2_{L^2(D)} + \|u^*\|_{L^2(D)}. \tag{5.91}$$

By observing that $(\mathcal{E}_T^d)^2$ is the quadrature approximation of $\|r_d\|^2_{L^2(D')}$ and $(\mathcal{E}_T^{int})^2$ is the quadrature approximation of $C_{r_{int}}$ with the quadrature rule (2.19), we can directly apply (2.20) to obtain the desired inequality (5.88). □

Remark 5.6.3. *The bound* (5.88) *is more complicated than the corresponding generalization error estimates for the Poisson, Heat and Wave equations. Nevertheless, the same general structure can be observed. There are two types of terms, one based on the training errors $\mathcal{E}_T^{int}, \mathcal{E}_T^{d}$, which can be computed a posteriori and should be made small during the training process. The other set of terms are decreasing in the number N_{int}, N_d of training samples. Thus the total error can be reduced by increasing the number of training points, while keeping the resulting training errors low.*

Remark 5.6.4. *We observe that the generalization error in* (5.88) *is given in terms of balls. Arbitrary sets can be readily included by considering the estimate in a union of balls containing that set. Moreover, the generalization error* (5.86) *only considers errors in the velocity field. In principle, this estimate can be used to bound pressure errors by working with the corresponding pressure Poisson equation.*

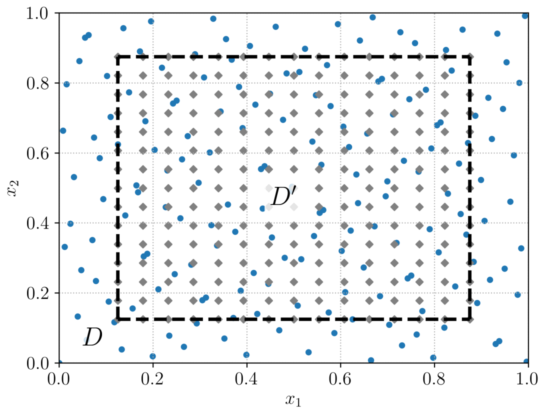


Figure 5.1: The domains D, D' for the Poission's equation numerical experiment. Training set $\mathcal{S}_{int}$ are Sobol points (blue dots) and training set $\mathcal{S}_d$ are Cartesian grid points (grey squares).

5.7 Numerical experiments

5.7.1 Poisson's Equation

We follow [75] and repeat their experiment from section 7.4.3. As in [75], the unique continuation problem for the Poisson equation is solved by considering the PDE (5.29) in the domain $D = (0, 1)^2$ (see figure 5.1), with source term,

$$f(x_1, x_2) = -60\Big(x_1 - x_1^2 + x_2 - x_2^2\Big). \tag{5.92}$$

The data (5.31) will be given in the observation domain (see figure 5.1),

$$D' = \{(x_1, x_2) \in \mathbb{R}^2 : |x_1 - 0.5| < 0.375;\ |x_2 - 0.5| < 0.375\}. \tag{5.93}$$

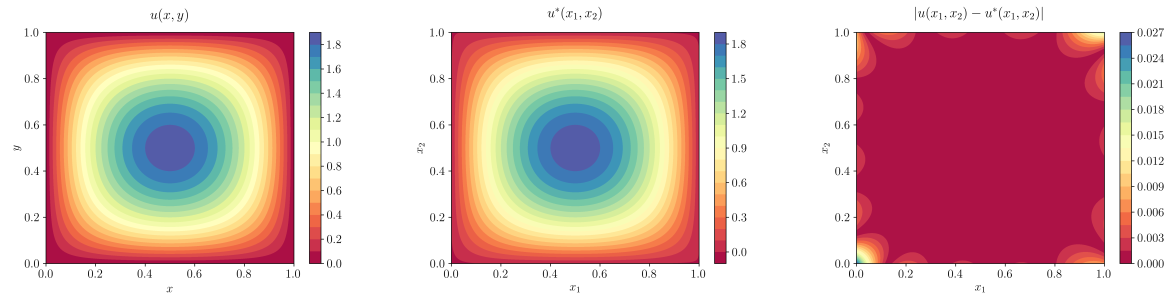
In order to generate the data term g in (5.31), we find that,

$$u(x_1, x_2) = 30x_1x_2(1 - x_1)(1 - x_2), \tag{5.94}$$

is an exact solution of the Poisson equation (5.29), with the source term (5.92) and let $g = u|_{D'}$, with D' (5.93), as the data term in (5.31). Clearly the exact solution u (5.94) and inputs f, g are as regular as required in Lemma 5.3.3.

N	$K-1$	$\bar{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\lvert\lvert u-u^*\rvert\rvert_{L^2}$	$\lvert\lvert u-u^*\rvert\rvert_{H^1}$
20×20	4	24	0.0	0.001	0.0008	0.28 %	1.1 %
40×40	4	24	0.0	0.001	0.0006	0.25 %	1.0 %
80×80	4	24	0.0	0.001	0.00053	0.24 %	0.9 %
160×160	4	24	0.0	0.001	0.00043	0.2 %	0.8 %

Table 5.1: Poisson's equation: relative percentage generalization errors and training errors for different numbers of training points.



(a) Exact Solution u (b) PINN u^* (c) Relative error $\frac{|\hat{u}|}{\|u\|_{L^2}}$

Figure 5.2: Comparison of the Exact solution, the PINN approximation and norm of the resulting error for the unique continuation problem for the Poisson's equation, with $N = 20 \times 20$ training points.

We will generate PINNs by algorithm 5, with loss functions (5.37), corresponding to training sets $\mathcal{S}_{int}, \mathcal{S}_d$. For $\mathcal{S}_{int}$, we will use Sobol points on the domain D (and the corresponding quasi-Monte Carlo integration as quadrature rule) and for $\mathcal{S}_d$, we set up a Cartesian grid on the inner square D' (5.93) (with midpoint rule as the quadrature rule), see figure 5.1 for a representation of both training sets.

The PINNs in algorithm 5 contain several hyperparameters namely, the number of hidden layers $K-1$ (depth) in (2.24), number of neurons $\bar{d}$ (width) in each hidden layer, the exponent q of the regularization term in loss function (3.6), the size λ_{reg} of the regularization term and the hyperparameter λ in the loss function (5.37). Therefore, consistently with the previous chapter, we follow [12] and perform an *ensemble training*, with the hyperparameter range specified in Table 3.1 to find the best hyperparameters for each experiment. To do so, for each hyperparameter configuration, we will run the BFGS-B optimizer with 30 randomly selected starting values and select the configuration resulting in the smallest training error $\mathcal{E}_T = \mathcal{E}_T^d + \mathcal{E}_T^{int}$ over the retrainings.

N	$K-1$	$\bar{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\|\|u-u^*\|\|_{L^2}$	$\|\|u-u^*\|\|_{H^1}$
20×20	4	24	0.0	0.001	0.0125	0.70 %	2.3 %
40×40	4	24	0.0	0.001	0.0127	0.53 %	1.9 %
80×80	4	24	0.0	0.001	0.0127	0.45 %	1.7 %
160×160	4	24	0.0	0.001	0.013	0.34 %	1.3 %

Table 5.2: Poisson's equation with noisy measurements: relative percentage generalization errors and training errors for different numbers of training points.

We set $r = \text{meas}(D') = 9/16$, with and for any $N = N_{int} + N_d$, we set $N_d = rN$ and present results for the PINNs in figure 5.2, where we plot the exact solution u (5.94), the PINN solution u^* and the error $\hat{u} = u^* - u$, associated with a PINN generated with $N = 20^2 = 400$ training points and with a network with 4 hidden layers, with 24 neurons in each layer, $\lambda = 10^{-3}$ in (5.37) and $\lambda_{reg} = 0$ in (3.6). We observe from this figure, that already for this very low number of training points, the PINN generated by algorithm 5 is able to approximate the underlying solution of the inverse problem (5.29), (5.31), very accurately. The plot of the error in Figure 5.2 (right) shows that the error is mostly concentrated near the boundary and shows a logarithmic behavior, as predicted by the theory and seen in [75].

The efficiency of PINNs in approximating solutions to this problem is reinforced from Table 5.1, where we present the (percentage relative) generalization errors $\mathcal{E}_G(D)$ (5.38) and total training errors $\mathcal{E}_T$, for PINNs with $N = 20^2, 40^2, 80^2, 160^2$, and consequently $N_d = rN$, training points. We also tabulate the (relative percentage) error in L^2 between the PINN and the exact solution. From the table, we see that even for very few (20^2) training points, the L^2 and H^1-generalization errors are very low, with H^1-error being around 1% and L^2-error around 0.3%. This is particularly impressive as it takes approximately $21s$ to train the PINN for this training set. The error decays slowly as the number of training samples is increased. This is consistent with the logarithmic decay predicted in (5.41). Moreover, the training error is already of the size of 10^{-4}, even for 20^2 training points and it is difficult to reduce the training error further.

Note that we present the generalization errors, averaged over $K = 30$ retrainings i.e, different random starting values for the optimizer. The generalization error, corresponding to the starting value with the smallest training loss, is considerably smaller.

Finally, as stated in remark 5.2.8, we can also extend the bounds (5.40), (5.41), to the case of noisy data, analogous to (5.28). To test the validity of PINNs in this regime, we perturb the right hand side of the data term (5.31), with 1% standard normal noise, run algorithm 5 to generate PINNs for approximating the inverse problem for the Poisson equation and present the results in Table 5.2. From this table, we see that although slightly higher than in the noise-free case (compare with Table 5.1), the L^2 and H^1-generalization errors are still very low and decay a bit faster than in the noise-free case. Thus, at least for noisy data with small noise amplitude, we can readily use PINNs for approximating the solutions of the data assimilation problem.

5.7.2 Heat equation

We follow [87] and start by considering the heat equation (5.46) in one space dimension (with zero Dirichlet boundary conditions) and setting $D = (0, 1)$. The final time is set as $T = 0.02$ and the data term in (5.47) is specified in the observation domain D'_T, with $D' = (a, 1 - a)$, and as in [87], we set $a = 0.2$. The data term g and the source term f are generated from the exact solution,

$$u(x, t) = e^{-4\pi^2 t^2} \sin(2\pi x). \tag{5.95}$$

In the first numerical experiment, we use Sobol points as training points $\mathcal{S}_{int}$ in the domain D_T and Cartesian grid points are chosen as the training set $\mathcal{S}_d \subset D'_T$ and $\mathcal{S}_{sb} \subset \partial D \times (0, T)$ (see figure 5.3 for an illustration). We run the algorithm 5 on these training sets to generate the PINN for approximating the data assimilation problem for the heat equation (5.46), (5.47).

In figure 5.4, we plot the exact solution and the solution field, generated by a PINN, and the relative error (in norm), with a total of $N = N_{int} + N_{sb} + N_d = 16 \times 50$ training points, with $N_{int} = (1 - r)N$, and $r = 0.6$ training points and with hyperparameters, shown in Table 5.3. We see from this figure that the PINN is able to approximate the exact solution (in both space and time) to very high accuracy. From the plot of the error (figure 5.4 (right)), we see that bulk of the error is concentrated near the initial time i.e $T \approx 0$. This is not surprising as the solution of the heat equation is damped very quickly in time and initial errors are dissipated.

To further quantity this high accuracy of PINNs, we present the percentage relative errors, $\|u - u^*\|_{L^2(D_T)}$ and $\|u - u^*\|_{L^2((0,T);H^1(D))}$, for a sequence of PINNs, with increasing number of training points. Note that this quantity is a slight perturbation of the generalization error in (5.57) and can be readily bounded above by the lhs of (5.57). For each configuration the model is retrained 20 times and the configuration realizing the lowest value of the average training loss over the retrainings is selected. We see from this table that the errors in both L^2 and H^1 are very small (significantly less than 1%), even for very few $16 \times 50 = 800$ training points. However, there is some saturation effect and the errors do not really decrease on increasing the number of training samples. This can be due to the fact that training error $\mathcal{E}_T$ is already very small, even for the smallest number of training points, and it is difficult to reduce them further with the LBFGS optimizer for the loss function (5.54).

N	$K-1$	$\bar{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\lVert u-u^*\rVert_{L^2}$	$\lVert u-u^*\rVert_{H^1}$
16×50	8	20	0.0	0.001	0.001	0.18 %	0.42 %
16×100	8	20	0.0	0.001	0.00096	0.25 %	0.52 %
16×200	8	20	0.0	0.001	0.00078	0.23 %	0.55 %

Table 5.3: 1-D Heat equation: relative percentages errors for different values of the number of training samples.

Finally, in order to investigate the sensitivity of the PINN errors with respect to the type of training points (underlying quadrature rule), we repeat the numerical experiment for data assimilation with the heat equation (with exact solution (5.95)), but with randomly chosen training points in the training sets $\mathcal{S}_{int,sb,d}$. All points are chosen with respect to underlying uniform distributions and the corresponding PINNs are generated by running algorithm 5. Note that we can readily prove a version of the generalization error estimate (5.57) in this setting by adapting the arguments in [73] (see Lemma 2.10) and also the recent paper [16]. The resulting (averaged over $K = 30$ different randomly chosen training sets) generalization errors in L^2 and H^1 are shown in Table 5.4. From this table, we observe that the generalization errors are as small as the ones in the case of Sobol and Cartesian training points. Moreover, there is a slight but consistent decay in the error with increasing number of training points. These results indicate considerable robustness of the PINNs, with respect to the choice of training points.

N	$K-1$	$\bar{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\lVert u-u^*\rVert_{L^2}$	$\lVert u-u^*\rVert_{H^1}$
16×50	4	24	0.0	0.001	0.001	0.27 %	0.55 %
16×100	4	24	0.0	0.001	0.00098	0.25 %	0.52 %
16×200	4	24	0.0	0.000788	0.00076	0.22 %	0.47 %

Table 5.4: 1-D Heat equation with randomly chosen training points: relative percentages errors for different values of the number of training samples.

N	$K-1$	$\bar{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\lVert u-u^*\rVert_{L^2}$
60×60	4	24	0.0	0.001	0.00125	0.29 %
90×90	4	20	0.0	0.001	0.0011	0.28 %
120×120	4	24	0.0	0.001	0.00085	0.21 %

Table 5.5: 1-D Wave equation with observation domain satisfying the geometric control condition and shown in figure 5.5(left): relative percentage L^2 errors for different values of the number of training samples.

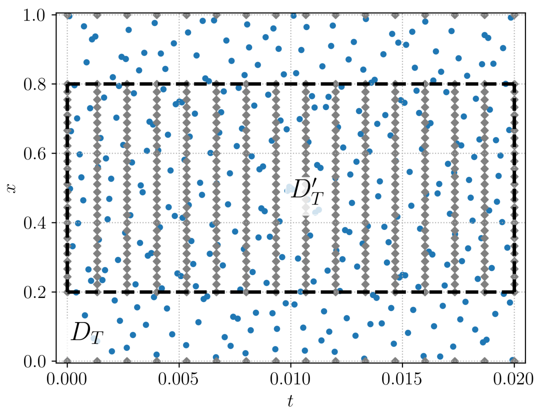
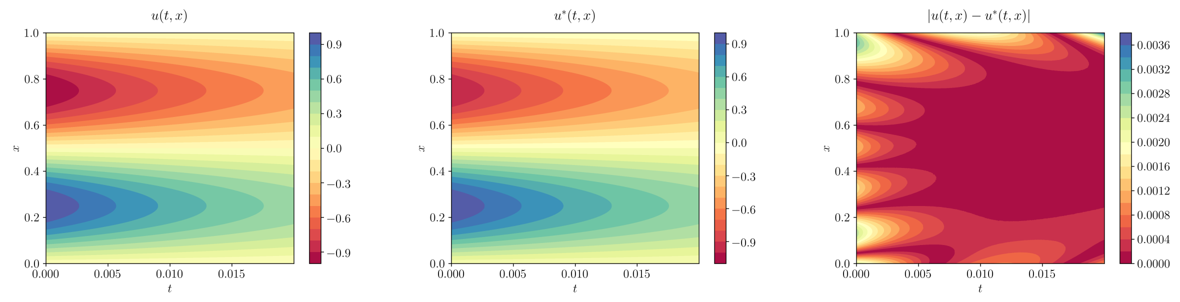


Figure 5.3: The domains D_T, D'_T for the heat equation numerical experiment. Training set $\mathcal{S}_{int}$ are Sobol points (blue dots) and training set $\mathcal{S}_d \cup \mathcal{S}_{sb}$ are Cartesian grid points (grey squares).



(a) Exact Solution u (b) PINN u^* (c) Relative error $\frac{|\hat{u}|}{\|u\|_{L^2}}$

Figure 5.4: Comparison of the Exact solution, the PINN approximation and relative error for the data assimilation problem for the heat equation, with $N = 16 \times 50$ training points.

5.7.3 Wave Equation

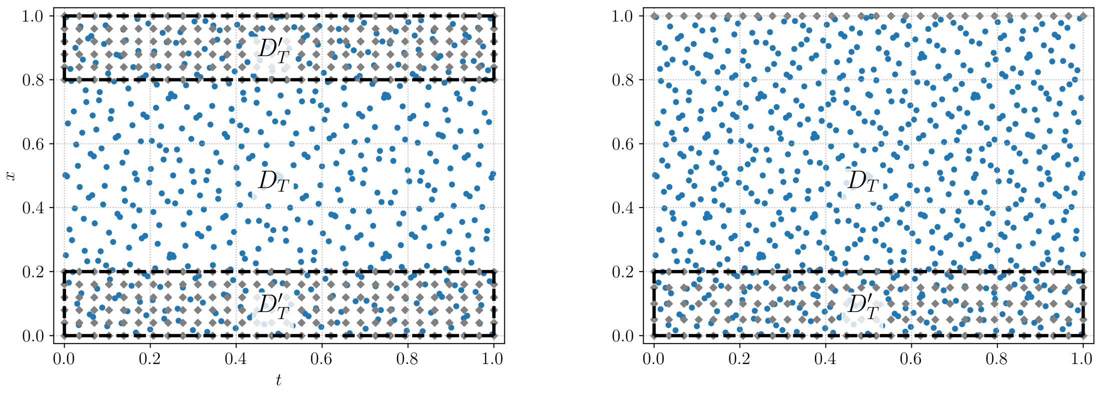
We present the following experiment proposed in [82], where the authors considered the wave equation in one space dimension, in the domain $D = [0, 1]$ and with final time $T = 1$. The source term f and data term g in (5.62) are generated from the exact solution

$$u(t, x) = \sin(2\pi t)\sin(2\pi x), \tag{5.96}$$

resulting in $f = 0$ and also satisfying the zero Dirichlet boundary conditions of (5.60).

For the first numerical experiment, we choose $D' = (0, 0.2) \cup (0.8, 1)$ as the domain on which data g is specified, see figure 5.5 for an illustration of the domains. Note that the resulting observation domain D'_T satisfies the geometric control condition [82]. For these domains, we choose training sets as follows: the training set $\mathcal{S}_{int}$ consists of Sobol points and the training sets $\mathcal{S}_{sb}$ and $\mathcal{S}_d$ are Cartesian grid points.

In figure 5.6, we plot the exact solution, the PINN and the resulting error, corresponding to $N = N_{int} + N_{sb} + N_d = 60 \times 60$ and $N_{int} = 0.6N$ training points. We see from this figure that the PINN approximates

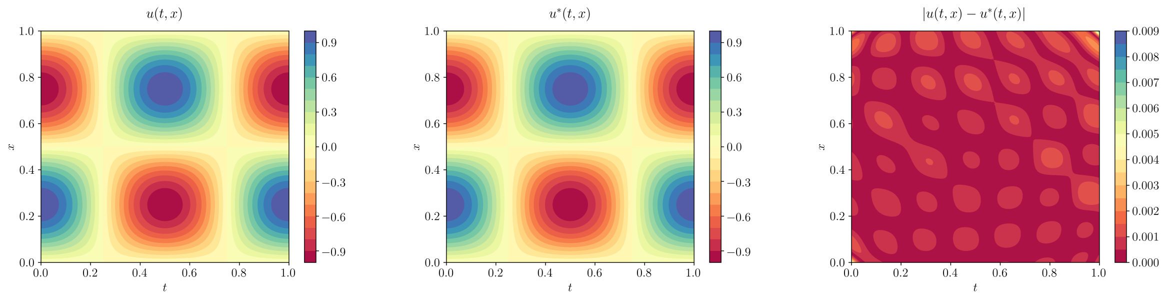


(a) Domain D_T' satisfying geometric control condition (b) Domain D_T' not satisfying geometric control condition

Figure 5.5: The domains D_T, D_T' for the numerical experiment for the wave equation. Left: Domain satisfying the geometric control condition (GCC), Right: Domain not satisfying the GCC. Training sets $\mathcal{S}_{int}$ are Sobol points (blue dots) and training set $\mathcal{S}_d \cup \mathcal{S}_{sb}$ are Cartesian grid points (grey squares).

the underlying solution very well, with errors being small and distributed throughout the space-time domain. The accuracy of the PINNs is further confirmed in Table 5.5, where we present the errors for a sequence of PINNs (with increasing number of training points). We focus on the $\sup\limits_{t\in[0,T]}\Big(\|u(\cdot,t) - u(\cdot,t)^*\|_{L^2(D)}\Big)$ error, which can be readily bounded above by the generalization error bound (5.70). We see from this table that the approximation errors are very low, with less than 0.3% relative error, already with 60×60 training points. This error decays further but saturates around a value of 0.2% for more training points.

For the second numerical experiment with the wave equation, we choose $D' = (0, 0.2)$ as the domain on which data g is specified. Note that this domain does not satisfy the geometric control condition [82]. The interior, spatial boundary and data training points are chosen similarly to the previous numerical experiment and are illustrated in figure 5.5 (right). In figure 5.8, we plot the exact solution, the PINN, and the corresponding error, obtained for $N = N_{int} + N_{sb} + N_d = 60 \times 60$, and $N_{int} = 0.8N$ training points. We see from this figure that although the geometric control condition is not satisfied, the PINN seems to approximate the underlying exact solution rather well. However, a close inspection of the error, plotted in figure 5.8 (right) reveals that the error is significantly greater than in the previous numerical experiment where the domain D_T' satisfied the geometric control condition. In particular, we see that the error seems to be transported along rays that do not belong to the observable part of the boundary. Nevertheless and as observed in [82], the errors are of small amplitude, even for this example. This is further verified in Table 5.6 where we present the generalization errors for this example for a sequence of PINNs, with increasing numbers of training points. From this table, we observe decay of the error with increasing number of training points. The overall generalization error is quite low, around 1.4% for largest number of training points considered here. Note that this error is still an order of magnitude larger than in the previous numerical experiment, where the geometric control condition was satisfied. This experiment nicely illustrates the role of the geometric control condition of [79] in this context.



(a) Exact Solution u (b) PINN u^* (c) Relative error $\frac{|\hat{u}|}{\|u\|_{L^2}}$

Figure 5.6: Data assimilation problem for the Wave equation in domains shown in figure 5.5 (left) satisfying the geometric control condition. Exact solution, PINN with $N = 60 \times 60$ training points and error.

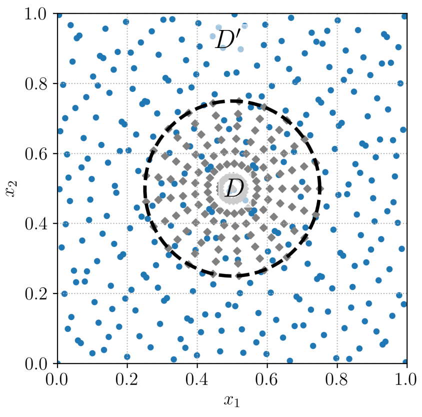


Figure 5.7: The domains D, D' for the numerical experiment for the Stokes equation. Training set $\mathcal{S}_{int}$ are Sobol points (blue dots) and training set $\mathcal{S}_d$ are Radial (Cartesian) grid points (grey squares).

5.7.4 Stokes Equation

We follow [84] and consider the homogeneous version of Stokes equation i.e $f, f_d \equiv 0$, in the two-dimensional domain $(0,1)^2$. We consider the exact solutions,

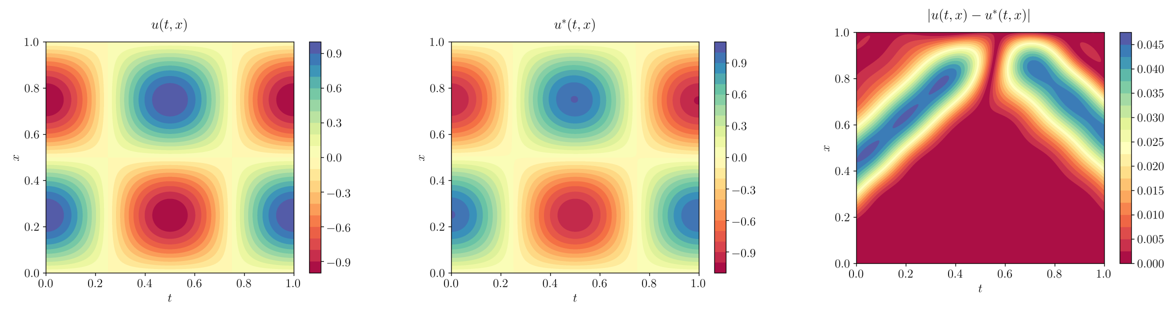
$$u(x_1,x_2) = (4x_1x_2^3, x_1^4 - x_2^4), \quad p(x_1,x_2) = 12x_1^2x_2 - 4x_2^3 - 1, \tag{5.97}$$

and define the data term $g = u|_{D'}$ on the sub-domain,

$$D' = \{(x_1,x_2) \in \mathbb{R}^2 : \sqrt{(x_1-0.5)^2 + (x_2-0.5)^2} < 0.25\}. \tag{5.98}$$

The domain D and the sub-domain D' are illustrated in figure 5.7. We chose Sobol points as the training set $\mathcal{S}_{int}$ and equally spaced (Cartesian) radial points as the training set $\mathcal{S}_d \subset D'$. See figure 5.7 for an illustration of these training sets.

With these training sets, we run algorithm 5 with loss function (5.85) to obtain PINNs approximating the data assimilation problem for the Stokes equation (5.73), (5.74). The results with a PINN, trained



(a) Exact Solution u (b) PINN u^* (c) Relative error $\frac{|\hat{u}|}{\|u\|_{L^2}}$

Figure 5.8: Data assimilation problem for the Wave equation in domains shown in figure 5.5 (right) satisfying the geometric control condition. Exact solution, PINN with $N = 60 \times 60$ training points and error.

N	$K-1$	$\bar{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\|\|u - u^*\|\|_{L^2}$
60×60	4	24	0.0	0.001	0.0011	2.2 %
90×90	4	24	0.0	0.001	0.00087	1.5 %
120×120	4	24	0.0	0.001	0.00081	1.4 %

Table 5.6: 1-D Wave equation with observation domain not satisfying the geometric control condition and shown in figure 5.5 (right): relative percentage L^2 errors for different values of the number of training samples.

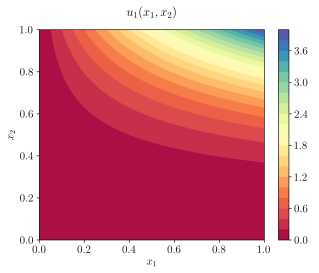
with $N = 40 \times 40$ and $N_d = rN$ points, with $r = \text{meas}(D')/\text{meas}(D)$, are shown in figure 5.9, where we plot the exact and PINNs solutions for the velocity components u_1, u_2 and pressure p. We see from this figure that the PINN is approximating the exact solution quite well, at least to the eye.

In order to quantify the approximation abilities of PINN for this example, in Table 5.7, we present the training errors and relative percentage errors of $\|u - u^*\|_{L^2(D)}$ and $\|p - p^*\|_{L^2(D)}$. From this table, we observe a slow, yet consistent, decay of the L^2-errors of the velocity field. This is consistent with the estimate (5.88) on the generalization error. More surprisingly, we also observe from the Table 5.7 that the pressure errors are reasonably small and decay with increasing numbers of training samples. Note that we did not provide any estimate on pressure errors, nor was pressure observed in the data in (5.74). Yet, given that pressure is a Lagrange multiplier for the Stokes equations and we can control pressure in terms of the velocity through the pressure Poission equations, we observe good approximation for the pressure by PINNs. If the pressure was also specified in the observation domain D', the data residual (5.84), and the loss function (5.85) can de readily modified and the resulting PINN is likely to lead to a better approximation of the pressure.

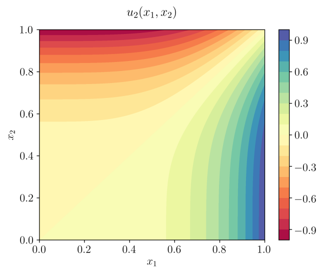
However, the amplitude of the error, for both the velocity field and the pressure, is higher than the other three equations that we considered earlier in the paper. This is not unexpected as the observation domain D' is smaller in this case (see figure 5.7) and has approximately 20% of the area of the whole domain D. To be able to reconstruct the velocity and pressure fields with reasonable accuracy from such a small observation domain is significant. Note that we have used the best retraining i.e, the PINN with the smallest training loss among the different random initializations of the optimizer. Choosing the average over all the retrainings led to slightly greater generalization error.

N	$K-1$	$\bar{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\|\|u-u^*\|\|_{L^2}$	$\|\|p-p^*\|\|_{L^2}$
20×20	4	24	0.0	0.001	0.0007	2.3 %	5.6 %
40×40	4	24	0.0	0.001	0.0004	1.7 %	4.0 %
80×80	4	20	0.0	0.01	0.00046	1.6 %	3.5 %

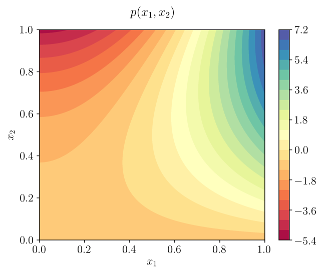
Table 5.7: Stokes equation: Relative percentages errors for different numbers of training points.



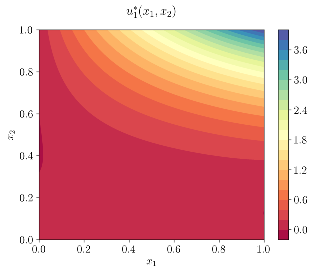
(a) Exact solution u_1



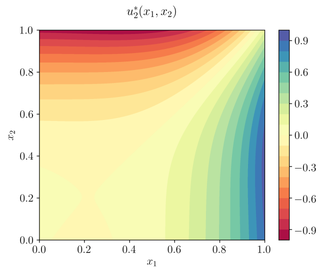
(b) Exact solution u_2



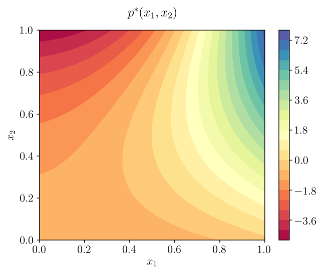
(c) Exact solution p



(d) PINN u_1^*



(e) PINN u_2^*



(f) PINN p^*

Figure 5.9: Comparison of the Exact solution and the PINN approximations for the inverse problem for the Stokes equation, with $N = 20\times 20$ training points.

6 Physics Informed Neural Networks for High-Dimensional PDEs

Chapter 3 and Chapter 5 provide a rigorous estimate of the generalization error for physics-informed neural networks (PINNs) when approximating forward and inverse problems of partial differential equations (PDEs). This estimate is derived based on the stability of the underlying problem. In summary, for a stable (forward or inverse) problem, the generalization error can be bounded by the following expression:

$$\mathcal{E}_G \leq C(u, u^*) \left(\mathcal{E}_T + C_{quad}(u, u^*, d) f(N, d)\right) \tag{6.1}$$

where $u, u^* \in \mathcal{U}(D_T)$, $D_T \subset \mathbb{R}^d$ represents the d-dimensional domain, C_{quad} and C are constants depending on the quadrature rule and the regularity of the solution, and f is a function of the number of training points.

It should be noted that the constants C_{quad} and the function f explicitly depend on the dimensionality of the problem. This is particularly true for standard composite quadrature rules and Quasi-Monte Carlo methods, as discussed in Section 2.4. On the other hand, Monte Carlo quadrature is independent of the problem's dimensionality.

In this regard, several works have rigorously demonstrated that PINNs can overcome the *curve of dimensionality*, when uniform distributed random training samples are used [73, 88], making PINNs a very attractive framework to solve high-dimensional PDEs. This is in contrast to traditional grid-based numerical methods such as finite elements or finite differences, which require approximately N^d degrees of freedom (for d dimensions, with N being the number of points in each dimension).

Therefore, in spite of their well-documented successes, it is clear that these methods are inadequate for approximating solutions of PDEs with high-dimensional state or parameter spaces. Such problems arise in different contexts ranging from PDEs with considerably large *state-space*, such as the Boltzmann, Radiative transfer, Schrödinger and Black-Scholes type equations with very high number of spatial dimensions, to many-query problems, as in uncertainty quantification (UQ), optimal design and inverse problems, which are modelled by PDEs with very high *parametric dimensions* [89, 90].

In this chapter, we first start with a motivating example of an high-dimensional *parametric PDE* (similar to the problem shown in 3.7.5) and afterwards focus on two major classes of high-dimensional PDEs, namely the radiative transfer equation and the Kolmogorov equations, which include many PDEs used in finance, such as the Black-Scholes equation.

6.1 A Motivating Example

As a first example, we consider the following parametric heat equation in the domain $\Omega = D_T \times P$:

$$u_t(x, t, \mu) = u_{xx}(x, t, \mu), \quad (x, t, \mu) \in \Omega, \tag{6.2}$$

with $D_T = D \times [0,T]$ ($T = 0.1$, $D = [-1,1]$) being the space-time domain, and $P = [0,1]^d$ the parameter-space.

The parametric nature of the equation arises from the initial condition which is defined as:

$$u(x,0,\mu) = \bar{u}(x,\mu) = \sum_{m=1}^{d} \bar{u}_m(x,\mu_m) = \sum_{m=1}^{d} -\frac{1}{d}\sin\left(\pi m \left(x-\mu_m\right)\right)/m^2. \tag{6.3}$$

We further consider Dirichlet boundary conditions $u(x,t,\mu) = u_b(x,t,\mu)$, $\forall (x,t,\mu) \in \partial D \times (0,T] \times P$ (obtained from the available exact solution of the PDE) and solve the underlying problem with PINNs. To this end, we take the following ansatz function:

$$u_\theta(x,t,\mu) = \sum_{m=1}^{d} \left(1 - \frac{t}{T}\hat{u}_{\theta,m}(x,t,\mu)\right) \bar{u}_m(x,\mu_m) \tag{6.4}$$

with

$$\hat{u}_\theta : \Omega \to \mathbb{R}^d \tag{6.5}$$

being a neural network of the form (2.24), with tunable parameters θ and with d outputs $\hat{u}_{\theta,1}, \hat{u}_{\theta,2}, ..., \hat{u}_{\theta,d}$. This ansatz for the solution is dictated by the following facts:

1. The heat equation is a linear PDE. Hence, the solution of the equation is the superposition of the solutions of the same PDE, each obtained with initial conditions $\bar{u}_m(x,\mu_m)$.
2. The solution of the heat equation shows temporal decay, which in the approximate solution is modelled by the multiplicative term $\left(1 - \frac{t}{T}\hat{u}_{\theta,m}(x,t,\mu)\right)$.
3. The ansatz directly enforces the initial condition.

Next, we define the main building blocks of algorithm 1.

Residuals.

- Interior Residual given by,

$$r_{int}[u_\theta](x,t,\mu) := \partial_t u_\theta(x,t,\mu) - \Delta_x u_\theta(x,t,\mu), \quad (x,t,\mu) \in D \times [0,T] \times P \tag{6.6}$$

- Spatial boundary Residual given by,

$$r_{sb}[u_\theta](x,t,\mu) := u_\theta(x,t,\mu) - u_b(x,t,\mu), \quad \forall (x,t,\mu) \in \partial D \times (0,T] \times P. \tag{6.7}$$

- Temporal boundary Residual given by,

$$r_{tb}[u_\theta](x,\mu) := u_\theta(x,0,\mu) - \bar{u}(x,\mu), \quad \forall (x,\mu) \in D \times P. \tag{6.8}$$

Given the definition of the the approximate ansatz solution (6.4), the temporal boundary residual $r_{tb}[u_\theta](x,t,\mu)$ is identically zero.

Training set. Given the potential high-dimensionality of the PDE, we choose the training sets based on uniformly distributed random points. Since, the temporal boundary residual is identically zero, we only need to define the interior training points $\mathcal{S}_{int} = \{y_m\}$ for $1 \leq m \leq N_{int}$, with each $y_m = (x_m, t_m, \mu_m) \in \Omega$, and spatial boundary training points $\mathcal{S}_{sb} = \{z_m\}$ for $1 \leq n \leq N_{sb}$, with each $z_n = (x_n, t_n, \mu_n) \in \partial D \times [0,T] \times P$

Loss function. According to the definitions above, the loss function is

$$J(\theta) := \sum_{m=1}^{N_{int}} w_n^{int} |r_{int}[u_\theta](x_m, t_m, \mu_m)|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb} |r_{sb}[u_\theta](x_n, t_n, \mu_n)|^2, \tag{6.9}$$

with $w_m^{int} = \frac{1}{N_{int}}$, $\forall m = 1, ..., N_{int}$, and $w_n^{sb} = \frac{1}{N_{sb}}$, $\forall n = 1, ..., N_{sb}$ being the quadrature weights corresponding to the interior and spatial boundary training points.

Given the problem at hand, we run algorithm 1 to approximate the solution of the heat equation for different numbers of input dimensions $d = [4, 10, 20, 40, 60, 80, 100]$. To do so, we set $\hat{u}_\theta : \Omega \to \mathbb{R}^d$ to be a neural network with 2 hidden layers and 20 neurons. For each d, we then train the model on a training set $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb}$ with $N_{int} = 16384$ and $N_{sb} = 8192$ training points, by minimizing the loss (6.9) for 10 different initializations of the trainable parameters, and pick the one minimizing the training loss. The resulting best-performing network is denoted as $u^*(x, t, \mu)$. The main metrics of interest are the generalization error:

$$\mathcal{E}_G = \left(\int_{D \times [0,T] \times P} |u(x, t, \mu) - u^*(x, t, \mu)|^2 dx dt d\mu \right)^{\frac{1}{2}} \tag{6.10}$$

and the elapsed training time. In this case an exact solution is readily available and given by

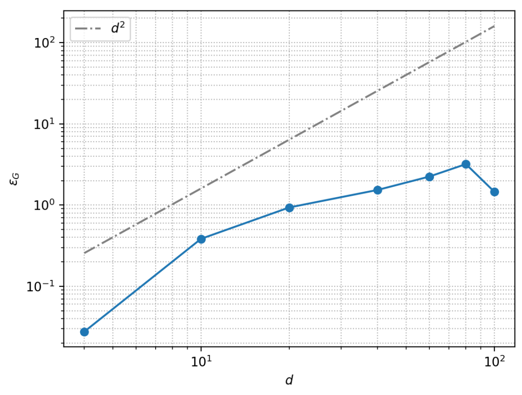
$$u(x, t, \mu) = \sum_{m=1}^{d} \exp\left(-(m\pi)^2 t\right) \bar{u}_m(x, \mu). \tag{6.11}$$

In Figure 6.1, we plot the achieved generalization error and the training time versus the number of dimensions d for the best performing model. From the plot, we observe that both the generalization error and the training time scale *polynomially* with the number of dimensions d. Specifically, the growth of the generalization error is approximately quadratic, whereas the growth of the training time is linear. This proves, at least experimentally, that PINNs can overcome the curse of dimensionality, which instead would imply an exponential increase of the metrics. Moreover, it should be noted that even for 100 dimensions, the final error is below 2%, and the surrogate model is obtained in only 2 hours.

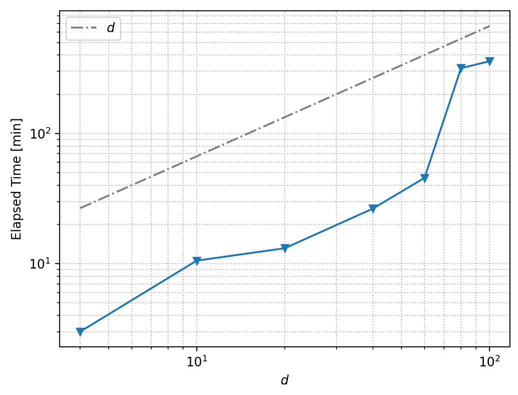
Finally, as an alternative to the physics-informed loss, given the low-dimensionality of the PDE state space (two dimensions, time and space), the model can be trained in a supervised fashion, based on a training set $\mathcal{S} = \{(x_i, t_j, \mu_k, u(x_i, t_j, \mu_k))\}$, $i = 1, ..., N$, $j = 1, ..., N_T$, $k = 1, ..., M$ generated by solving equation (6.4) with standard numerical methods, on a spatial grid with spatial resolution $\Delta x := x_{i+1} - x_i$, $\forall i = 1, \ldots, N$, for N_T time-steps and for M realizations of the parameters $\mu_k \sim \text{Unif}(P)$, $k = 1, \ldots, M$. However, for problems with high-dimensional state space, solving the PDE even once might entail an extremely large computational cost. We will address PDEs belonging to this class in the following sections.

6.2 Radiative Transfer Equation

The study of radiative transfer is of vital importance in many fields of science and engineering including astrophysics, climate dynamics, meteorology, nuclear engineering, and medical imaging [91]. The fundamental equation describing radiative transfer is a *linear partial integro-differential equation*, termed as



(a) Generalization error VS number of dimensions d



(b) Elapsed Training time VS number of dimensions d

Figure 6.1: Relative percentage generalization error (left) and training time (right) of PINN trained with loss (6.9) versus the number of dimensions d of the underlying parameter space.

the *radiative transfer equation*. Under the assumption of a static underlying medium, it has the following form [91],

$$\frac{1}{c}u_t + \omega \cdot \nabla_x u + ku + \sigma\left(u - \frac{1}{s_d}\int\limits_{\Lambda}\int\limits_{S} \Phi(\omega,\omega',\nu,\nu')u(x,t,\omega',\nu')d\omega' d\nu'\right) = f, \tag{6.12}$$

with time variable $t \in [0,T]$, space variable $x \in D \subset \mathbb{R}^d$ (and $D_T = D \times [0,T]$), *angle* $\omega \in S = \mathbb{S}^{d-1}$ i.e. the d-dimensional sphere and *frequency* (or group energy) $\nu \in \Lambda \subset \mathbb{R}$. The constants in (6.12) are the speed of light c and the surface area s_d of the d-dimensional unit sphere. The unknown of interest in (6.12) is the so-called *radiative intensity* $u : D_T \times S \times \Lambda \to \mathbb{R}$, while $k = k(x,\nu) : D \times \Lambda \to \mathbb{R}_+$ is the *absorption coefficient* and $\sigma = \sigma(x,\nu) : D \times \Lambda \to \mathbb{R}_+$ is the *scattering coefficient*. The integral term in (6.12) involves the so-called *scattering kernel* $\Phi : S \times S \times \Lambda \times \Lambda \to \mathbb{R}$, which is normalized as

$$\int_{S\times\Lambda} \Phi(\cdot,\ \omega',\ \cdot,\ \nu')d\omega' d\nu' = 1,$$

in order to account for the conservation of photons during scattering. The dynamics of radiative transfer are driven by a source (emission) term $f = f(x,\nu) : D \times \Lambda \to \mathbb{R}$.

Although the radiative transfer equation (6.12) is linear, explicit solution formulas are only available in very special cases [91]. Hence, numerical methods are essential for the simulation of the radiative intensity in (6.12). However, the design of efficient numerical methods is considered to be very challenging, on account of the *high-dimensionality* of the radiative transfer equation (6.12), where in the most general case of three space dimensions, the radiative intensity is a function of 7 variables (4 for space-time, 2 for angle and 1 for frequency) [89, 90, 91]. In order to solve the radiative transfer with PINNs, we apply algorithm 1. Below we explicitly define the building blocks of the algorithm.

6.2.1 The underlying PDEs

We model radiative transfer in a static medium by the evolution equation (6.12) for the radiative intensity u. This partial integro-differential equation is supplemented with the initial condition,

$$u(x, 0, \omega, \nu) = u_0(x, \omega, \nu), \quad (x, \omega, \nu) \in D \times S \times \Lambda, \tag{6.13}$$

for some initial datum $u_0 : D \times S \times \Lambda \to \mathbb{R}$.

Given that the radiative transfer equation (6.12) is a *transport equation*, the boundary conditions are imposed on the so-called *inflow boundary* given by,

$$\Gamma_- = \{(x, t, \omega, \nu) \in \partial D \times [0, T] \times S \times \Lambda : \omega \cdot n(x) < 0\} \tag{6.14}$$

with $n(x)$ denoting the unit outward normal at any point $x \in \partial D$ (the boundary of the spatial domain D). We specify the following boundary condition,

$$u(x, t, \omega, \nu) = u_b(x, t, \omega, \nu), \quad (x, t, \omega, \nu) \in \Gamma_-, \tag{6.15}$$

for some boundary datum $u_b : \Gamma_- \to \mathbb{R}$.

Given that the radiative intensity is a function of $2d + 1$ variables, it is essential to find suitable low-dimensional functionals (observables) to visualize and interpret it. To this end, one often considers physically interesting angular-moments such as the *incident radiation* (zeroth angular moment) and *heat flux* (first angular moment) given by,

$$G(x, t, \nu) = \int_S u(x, t, \omega, \nu) d\omega \tag{6.16}$$

$$F(x, t, \nu) = \int_S u(x, t, \omega, \nu) \omega d\omega \tag{6.17}$$

We note that for many applications of radiative transfer, it is common to consider the steady (time-independent) version of the radiative transfer equation (6.12), which formally results from setting $c \to \infty$ and dropping the time-derivative term in the left hand side of (6.12).

6.2.2 PINNs

Training sets

As in section 3.1, we divide the training set $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$ into the following three subsets,

- Interior training points $\mathcal{S}_{int} = \{z_j^{int}\}$, for $1 \leq j \leq N_{int}$, and $z_j^{int} = (x_j^{int}, t_j^{int}, \omega_j^{int}, \nu_j^{int})$ with $x_j^{int} \in D$, $t_j^{int} \in [0, T]$, $\omega_j^{int} \in S$, $\nu_j^{int} \in \Lambda$, for all j. These points are the quadrature points of a suitable quadrature rule with weights w_j^{int}. If the underlying spatial domain $D \subset \mathbb{R}^d$ can be mapped to a d-dimensional rectangle, either entirely or in patches, then we can set the training points z_j^{int} as a low-discrepancy Sobol sequence [92] in $[0, 1]^{2d+1}$, by rescaling the relevant domains. Sobol sequences arise in the context of Quasi-Monte Carlo integration [50] and the corresponding quadrature weights are $w_j^{int} \equiv \frac{1}{N_{int}}$, for all j. Note that the QMC quadrature rule does not suffer from the curse of dimensionality (see section 6.2.3 for details). In case the geometry of the domain is very complicated, one has simply choose random points, independent and identically distributed with the underlying uniform distribution, as training points.

- Spatial Boundary points $\mathcal{S}_{sb} = \{z_j^{sb}\}$, for $1 \leq j \leq N_{sb}$, and $z_j^{sb} = (x_j^{tb}, t_j^{tb}, \omega_j^{tb}, \nu_j^{tb})$ with $x_j^{tb} \in \partial D$, $t_j^{tb} \in [0,T]$, $\omega_j^{tb} \in S$, $\nu_j^{tb} \in \Lambda$, for all j. These points are the quadrature points of a suitable quadrature rule with weights w_j^{sb}. As before, we can choose Sobol points for logically rectangular domains D or random points.
- Temporal boundary points $\mathcal{S}_{tb} = \{z_j^{tb}\}$, for $1 \leq j \leq N_{tb}$, and $z_j^{tb} = (x_j^{tb},\ \omega_j^{tb},\ \nu_j^{tb})$ with $x_j^{tb} \in D, \omega_j^{tb} \in S, \nu_j^{tb} \in \Lambda$, for all j. These points are the quadrature points of a suitable quadrature rule with weights w_j^{tb}. We can choose Sobol points for logically rectangular domains D or random points to constitute this training set.

Residuals

Next, we follow Section 3.1 and define the following *PDE residual* $r_{int}[u_\theta] = r_{int}[u_\theta](x,t,\omega,\nu)$, for all $(x,t,\omega,\nu) \in D_T$,

$$r_{int}[u_\theta] := \frac{1}{c}\partial_t u_\theta + \omega \cdot \nabla_x u_\theta + k u_\theta + \sigma\left(u_\theta - \frac{1}{s_d}\sum_{i=1}^{N_S} w_i^S \Phi(\omega, \omega_i^S, \nu, \nu_i^S) u_\theta(x,t,\omega_i^S,\nu_i^S)\right) - f, \tag{6.18}$$

where we have used *Gauss-Legendre* quadrature rules [49] with quadrature points and corresponding weights $z_i^S = (\omega_i^S, \nu_i^S)$, for $1 \leq i \leq N_S$, and $\omega_i^S \in S$ and $\nu_i^S \in \Lambda$, to approximate the scattering kernel integral in (6.12). Moreover, k, σ, f are defined from (6.12).

We also need the following residuals for the initial and boundary conditions,

$$\begin{aligned} r_{tb}[u_\theta] &:= u_\theta - u_0, \qquad \forall (x,\omega,\nu) \in D \times S \times \Lambda, \\ r_{sb}[u_\theta] &:= u_\theta - u_b, \qquad \forall (x,t,\omega,\nu) \in \Gamma_-. \end{aligned} \tag{6.19}$$

Loss Functions

Based on the residuals and training sets above, we complete algorithm 1 by defining the following loss function,

$$J(\theta) := \sum_{j=1}^{N_{sb}} w_j^{sb} |r_{sb}[u_\theta](z_j^{sb})|^2 + \sum_{j=1}^{N_{tb}} w_j^{tb} |r_{tb}[u_\theta](z_j^{tb})|^2 + \lambda \sum_{j=1}^{N_{int}} w_j^{int} |r_{int}[u_\theta](z_j^{int})|^2. \tag{6.20}$$

6.2.3 Estimates on the generalization error

For the sake of definiteness and simplicity, we consider the spatial domain as $D = [0,1]^d$, with d being the spatial dimension. Any rectangular domain $\prod_{i=1}^{d} [a_i, b_i]$, with $a_i < b_i$, for any $a_i, b_i \in \mathbb{R}$ can be mapped to $[0,1]^d$ by rescaling. Similarly, logically (patch or block) cartesian domains can be transformed to $(0,1)^d$ by combinations of coordinate transforms. We also rescale time and frequency to set $T = 1$ and $\Lambda = [0,1]$. Finally, the angular domains can be mapped onto to $[0,1]^{d-1}$ by rescaling the underlying polar coordinates. Hence, the underlying domain is $D_T = [0,1]^{2d+1}$. Thus, we can choose our interior training points $\mathcal{S}_{int}$, temporal boundary training points $\mathcal{S}_{tb}$ and spatial boundary training points $\mathcal{S}_{sb}$ as low-discrepancy Sobol points [50].

Our aim in this section is to derive a rigorous estimate on the so-called *generalization error* (or approximation error) for the trained neural network $u^* = u_{\theta^*}$, which is the output of the PINNs algorithm 1. This error is of the form,

$$\mathcal{E}_G = \mathcal{E}_G(\theta^*) := \left(\int\limits_{D_T} |u(x,t,\omega,\nu) - u^*(x,t,\omega,\nu)|^2 dz \right)^{\frac{1}{2}}, \tag{6.21}$$

with $dz = dxdtd\omega d\nu$ denoting the volume measure on D_T.

We follow Chapter 3.1, Section 3.2 and estimate the generalization error (6.21), in terms of *training errors*,

$$\mathcal{E}_T^{sb} := \left(\sum_{j=1}^{N_{sb}} w_j^{sb} |r_{sb}[u_{\theta^*}](z_j^{sb})|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_T^{tb} := \left(\sum_{j=1}^{N_{tb}} w_j^{tb} |r_{tb}[u_{\theta^*}](z_j^{tb})|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_T^{int} := \left(\sum_{j=1}^{N_{int}} w_j^{int} |r_{int}[u_{\theta^*}](z_j^{int})|^2 \right)^{\frac{1}{2}} \tag{6.22}$$

Note that the training errors, defined above, correspond to a local minimizer θ^* of (3.6) and are readily computable from the loss function (3.6), during and at the end of the training process.

The detailed estimate on the generalization error in Lemma 9.1.1, together with the assumptions on the underlying coefficients, functions and neural network, is presented and proved in Appendix 9.1.1. We direct the interested reader to the appendix and focus on the following form of the error estimate (9.3),

$$\begin{aligned}(\mathcal{E}_G)^2 &\leq C_1 \left((\mathcal{E}_T^{tb})^2 + c(\mathcal{E}_T^{sb})^2 + c(\mathcal{E}_T^{int})^2 \right) \\ &+ C_2 \left(\frac{(\log(N_{tb}))^{2d}}{N_{tb}} + c\frac{(\log(N_{sb}))^{2d}}{N_{sb}} + c\frac{(\log(N_{int}))^{2d+1}}{N_{int}} + cN_S^{-2s} \right),\end{aligned} \tag{6.23}$$

with finite constants $C_1 = C$, $C_2 = CC^*$ defined in (9.4). The following remarks about the bound (6.23) are in order,

Remark 6.2.1. *We see from the right hand side of the bound* (6.23) *that the dimensional dependence of the upper bound is only a logarithmic factor. This is not a severe restriction in this case, as the spatial dimension d is atmost* 3. *It is well known [50] that the logarithmic factor in the rhs of* (6.23) *starts affecting the rate of decay only when* $N_{int} < 2^{2d+1}$. *Thus as long as* $N_{int} > 128$ *and* $N_{tb}, N_{tb} > 64$, *we should see a linear decay in the error contributions of the Sobol points in* (6.23). *Hence, we claim that as long as the training errors do not depend on the underlying dimension, the estimate* (6.23) *suggests that the PINNs algorithm 1 will* not suffer from a curse of dimensionality.

Remark 6.2.2. *The estimate* (6.23) *brings out the role of the speed of light c very clearly. As long as c is finite, we can rescale time to set* $c = 1$. *Nevertheless, the constant* C_1 *in* (6.23) *grows exponentially with the rescaled time, deteriorating the control on the error provided by the bound* (6.23). *Thus, this bound is not suitable for steady-state problems (formally) obtained by letting* $c \to \infty$. *Nevertheless, a modified error estimate can be derived for the steady state case and we present it in the appendix 9.1.1.*

6.2.4 Numerical Experiments

The PINNs algorithm has the following hyperparameters, the number of hidden layers $K-1$, the width of each hidden layer $d_k \equiv \bar{d}$ in (2.24), the specific activation function A, the parameter λ in the loss function

(6.20), the regularization parameter λ_{reg} in the cumulative loss function (3.6) and the specific gradient descent algorithm for approximating the optimization problem (3.6). We use the hyperbolic tangent tanh activation function, thus ensuring that all the smoothness hypothesis on the resulting neural networks, as required in lemmas 9.1.1 and 9.1.2 are satisfied. Moreover, we use the second-order LBFGS method [56] as the optimizer. We follow the ensemble training procedure of [12] in order to choose the remaining hyperparameters. To this end, we consider a range of values, shown in Table 6.1, for the number of hidden layers, the depth of each hidden layer, the parameter λ and the regularization parameter λ_{reg}. For each configuration in the ensemble, the resulting model is retrained (in parallel) n_θ times with different random starting values of the trainable weights in the optimization algorithm and the one yielding the smallest value of the training loss is selected.

	$K-1$	$\bar{d}$	λ	λ_{reg}	n_θ
Example 6.2.4, 6.2.4	4, 8	16, 20, 24	0.1, 1, 10	0	5
Example 6.2.4	4, 8	16, 20	0.1, 1	0, 10^{-6}, 10^{-5}	10
Example 6.2.4	4, 8, 12, 16, 20	16, 20, 24, 28, 32, 36, 40	0.1, 1	0	20
Example 6.2.4	4, 8	16, 20, 24	1, 10	0	5

Table 6.1: Hyperparameter configurations and number of retrainings employed in the ensemble training of PINNs for the radiative transfer equation (6.12)

Monochromatic stationary radiative transfer in one space dimension

We begin with the much simpler case of steady state radiative transfer in the one space dimension, also referred to as slab geometry [93]. In this case, the radiative transfer equations (6.12) simplify to,

$$\mu\frac{\partial}{\partial x}u(x,\mu) + \Big(\sigma(x)+k(x)\Big)u(x,\mu) = \frac{\sigma(x)}{2}\int_{-1}^{1}\Phi(\mu,\mu')u(x,\mu')d\mu', \quad \mu=\cos(\theta), \quad (x,\mu)\in[0,1]\times[-1,1]. \tag{6.24}$$

We follow the setup of [94] where the authors benchmarked least squares finite element methods for one-dimensional radiative transfer on this problem. As in [94], the following *inflow* boundary conditions are imposed:

$$\begin{aligned} u(0,\mu) &= 1, \quad \mu\in(0,1], \\ u(1,\mu) &= 0, \quad \mu\in[-1,0). \end{aligned} \tag{6.25}$$

Note that the boundary conditions allow for possible discontinuities at $\mu = 0$. The coefficients and scattering kernel are,

$$\sigma(x)=x, \quad k(x)=0, \quad \Phi(\mu',\mu)=\sum_{\ell=0}^{L} d_\ell P_\ell(\mu)P_\ell(\mu'), \quad d_0=1, \tag{6.26}$$

with $P_\ell(\mu)$ denoting the Legendre polynomial of order ℓ. We employ the sequence of coefficients $d_\ell = \{1.0, 1.98398, 1.50823, 0.70075, 0.23489, 0.05133, 0.00760, 0.00048\}$, proposed in [94]. Although only in 2 dimensions, this problem is nevertheless considered rather challenging on account of the possible presence of discontinuities.

We use the PINNs algorithm 1 to approximate (6.24), with Sobol points for the interior training set $\mathcal{S}_{int}$ and spatial boundary training set $\mathcal{S}_{sb}$. Similarly, a Gauss-Legendre quadrature rule with $N_S = 10$

N_{int}	N_{sb}	$K-1$	$\bar{d}$	λ	$\mathcal{E}_T$	$\|\|u_- - u_-^*\|\|_{L^2}$	$\|\|u_+ - u_+^*\|\|_{L^2}$	$\|\|u - u^*\|\|_{L^2}$	Training Time
8192	2048	8	24	0.1	0.00015	1.1%	1.2%	0.24%	20 min

Table 6.2: Results for monochromatic stationary radiative transfer in one space dimension.

quadrature points is used for approximating the integral with the scattering kernel. We also set $N_{int} = 8192$, $N_{sb} = 2048$, for this experiment. The hyperparameters that resulted from the ensemble training are presented in Table 6.2. As seen from the table, a very low training error is obtained in this case, together with a comparably low generalization error of 0.24%. A contour plot of the resulting radiative intensity in (x, μ)-plane is presented in figure 6.2. The results are very similar to those obtained with a finite element solver. It is interesting to note that this very good match with the finite element method is obtained with a training time of 20 minutes on a CPU.

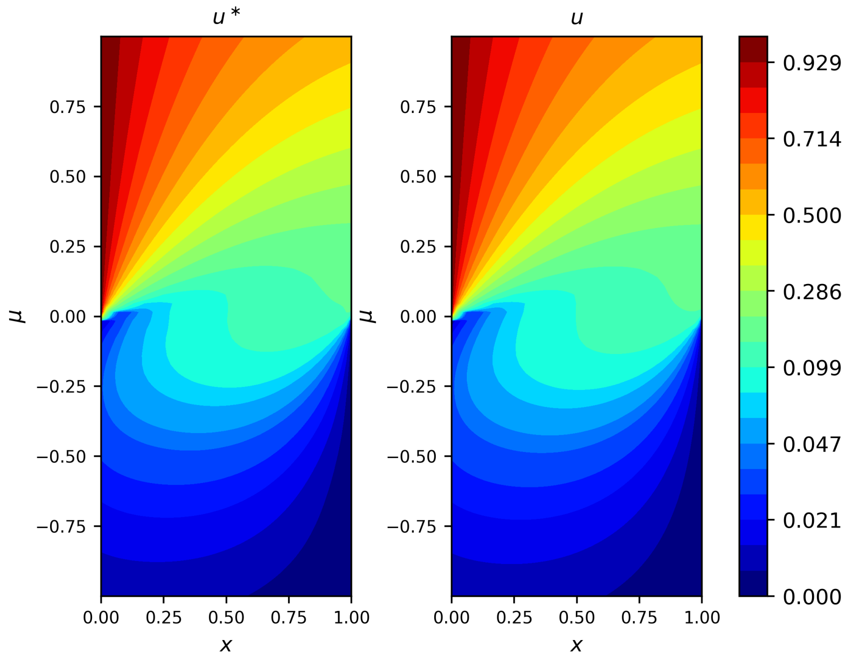


Figure 6.2: Contour plot of the PINN radiative intensity $u^*(x, \mu)$ for the 1D monochromatic experiment (left), compared with the solution $u(x, \mu)$ obtained with a finite element solver (right)

Another attractive feature of this simplified problem lies in the fact that the authors in [95] obtained an exact analytical solution for it. Although it is very complicated to evaluate this solution for the whole (x, μ)-plane, its values on the boundaries $u_-(\mu) = u(0, \mu)$ and $u_+(\mu) = u(1, \mu)$ can be readily evaluated. We do so and compare the exact solution with the trained PINN, denoted by $u_\pm^*$. These results are plotted in figure 6.3. We see from this figure that the PINN is able to very accurately approximate the discontinuous exact solution at the boundary. A quantitative comparison in performed by computed the errors $u_\pm - u_\pm^*$ in L^2-norm. These errors, presented in table 6.2, are very small for both boundaries and further demonstrate that the PINN is able to approximate the underlying discontinuous solution to high-accuracy, at very low computational cost.

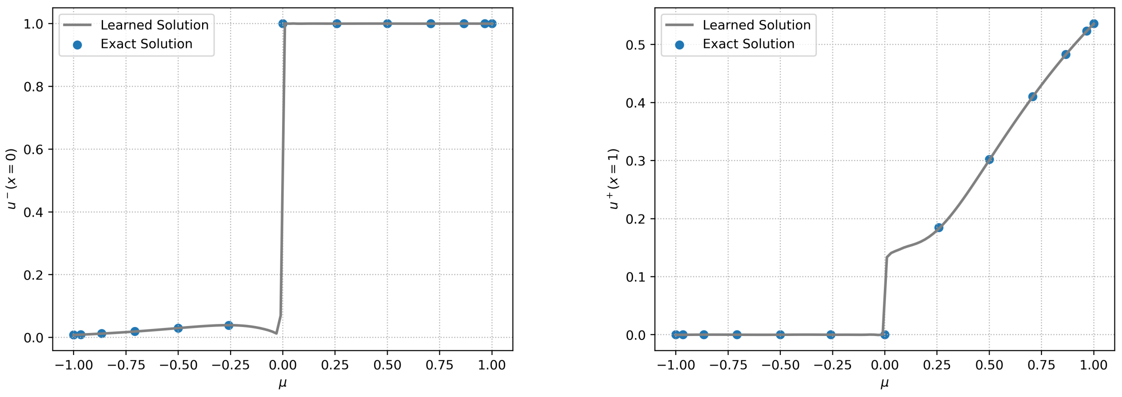


Figure 6.3: Comparison of the analytical and PINN radiative intensity at the physical domain boundaries for the stationary monochromatic radiative transfer in one-space dimension.

Monochromatic stationary radiative transfer in three space dimensions

Next, we consider a monochromatic and stationary version of the general radiative transfer equations (6.12), but in three space dimensions. Already, this problem is in 5 dimensions and is challenging on account of possibly high computational cost. We use the same setup as in [96] (section 8.2, experiment 3) and consider the problem in the unit cube $D = [0,1]^3$ where a source, located at the center $c = (0.5, 0.5, 0.5)$, radiates into the surrounding medium. We consider no further radiation entering the domain (zero Dirichlet boundary conditions on the inflow boundary). The source term f is given by

$$f(x) = k(x)I_b(x), \quad I_b(x) = \begin{cases} 0.5 - r, & r \leq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{6.27}$$

with $r = |x - c|$. The absorption coefficient is $k(x) = I_b(x)$ and isotropic scattering $\Phi = 1$, with unit scattering coefficient $\sigma(x) = 1$ is considered.

As before, we use Sobol points for the interior training set $\mathcal{S}_{int}$ and boundary training set $\mathcal{S}_{sb}$. Quadrature points, corresponding to a Gauss quadrature rule of order 20 are also used to approximate the scattering integral. We set $N_{int} = 16384$, $N_{sb} = 12288$ and $N_S = 100$. The hyperparameters, corresponding to the best performing networks, that result from ensemble training are presented in Table 6.3. We see from this table that this hyperparameter configuration resulted in a very low (total) training error of 4.4×10^{-4}, which is comparable to those obtained in the one-space dimension case (see table 6.2).

N_{int}	N_{sb}	$K-1$	$\bar{d}$	λ	$\mathcal{E}_T$	Training Time
16384	12288	8	24	0.1	0.00044	1 hr 9 min

Table 6.3: Results of the ensemble training for the stationary monochromatic radiative transfer in three space dimensions.

As there is no analytical solution available for the radiative intensity in this case, we cannot compute generalization errors. However, based on the theory (see estimate (9.3)) and on the comparison with the one-dimensional case, we expect very low generalization errors when the training errors are this low. Moreover, we can perform qualitative comparisons with the results obtained in [96]. To this end, we plot three-dimensional volume plot for the *incident radiation* $G(x)$ (see the first equation in (6.16) for definition) in figure 6.4. We see from this figure that the results with PINN are very similar to the results with the discrete ordinate method, shown in [96] (figure 8.12, page 126). Thus, we are able to approximate the incident radiation to the same accuracy as a discrete ordinate method. The main differences lies in the simplicity of implementation and very low computational cost. We observe from table 6.3 that the PINN was trained in approximately 70 minutes on a single GPU. This should be contrasted with the very intricate parallel algorithm of [96], which required considerably more computational time as the method resulted in very number of degrees of freedom ranging from $200000 - 600000$.

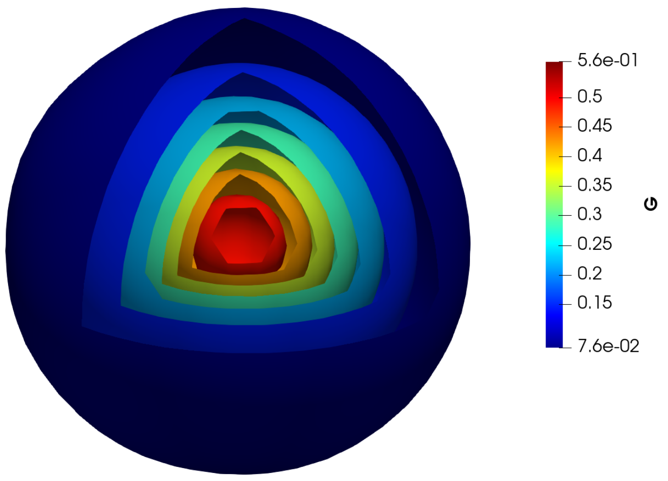


Figure 6.4: Contour plot of the incident radiation $G(x)$ for the 3D monochromatic experiment

Polychromatic stationary radiative transfer in three space dimensions

Next, we consider the most general case of the steady state radiative transfer equation (9.16) by following the setup of [97] and references therein, where (9.16) is considered in the unit cube $D = [0,1]^3$ and in the frequency domain $\Lambda = [-6, 6]$, with normalization of energy groups. Furthermore, we consider a simple case of zero absorption, isotropic kernel, zero Dirichlet boundary conditions (on the inflow boundary) and spherical symmetry. Under the assumptions, by integrating equation (9.16) over the unit sphere S, we arrive at the following ordinary differential equation for the radial flux i.e the incident heat flux (6.16) along the radius,

$$\nabla \cdot F_r = \frac{1}{r^2}\frac{d}{dr} r^2 F_r = 4\pi f(r, \nu) \tag{6.28}$$

with $r = |x - (0.5, 0.5, 0.5)|$ (see also [97] and references therein).

An exact solution for the above ODE can be easily obtained. In particular, with the source term:

$$f(x,\nu) = \begin{cases} \sqrt{\pi}\phi(\nu)\Big(1-2r\Big) & \text{if } r \leq 0.5, \\ 0 & \text{otherwise,} \end{cases} \qquad \phi(\nu) = \frac{1}{\sqrt{\pi}}\exp\big(-\nu^2\big), \tag{6.29}$$

the radial flux F_r results in

$$F_r = \begin{cases} 4\sqrt{\pi^3}\phi(\nu)\Big(\frac{r}{3}-\frac{r^2}{2}\Big) & \text{if } r \leq 0.5, \\ 4\sqrt{\pi^3}\phi(\nu)\frac{1}{96r^2} & \text{otherwise.} \end{cases} \tag{6.30}$$

As in the previous numerical experiment, we use Sobol points for the interior and boundary training sets and Gauss quadrature points for integrating the scattering kernel, with $N_{int} = 16384$, $N_{sb} = 12288$, and $N_S = 100$. The hyperparameters used in the ensemble trainig are reported in table 6.1 and the resulting best performing configuration is shown in table 6.4. We observe from this table that the resulting training error is 1.6×10^{-3}, which is about three times higher than the training error with the monochromatic experiment (see table 6.3). This is not surprising as the underlying problem is more complicated on account of introducing frequency as an additional variable and resulting in a 6-dimensional problem.

As no analytical solution is available for the radiative intensity, we cannot compute the generalization error (9.19). However, we can compute the error between the analytical radial flux (6.30) and the PINN approximation (computed from the intensity with a Gauss-Legendre quadrature rule). We show the resulting L^2-norm of the error in table 6.4. We see from this table that the error for the flux is quite low at approximately 2% relative error, even for this rather complicated underlying problem. Moreover, the training time is only one hour, comparable to the training time required to train the model for the previous experiment, despite including one additional input dimension compared to before.

N_{int}	N_{sb}	$K-1$	$\bar{d}$	λ	$\mathcal{E}_T$	$\lvert\lvert F_r - F_r^*\rvert\rvert_{L^2}$	Training Time
16384	12288	8	20	0.1	0.0016	2.1 %	1 hr 6 min

Table 6.4: Results for steady polychromatic radiative transfer in three space dimensions.

Polychromatic time-dependent Radiative transfer in three space dimensions

For the final numerical experiment, we consider the configuration proposed in [98], which is widely used in benchmarking the radiative transport modules in production codes for radiation-(magneto)hydrodynamics, in the context of Astrophysics [99]. The setup is as follows; a sphere with radius R_i and fixed temperature T_S is surrounded by a cold static medium at temperature $T_m < T_S$. The experiment might represent, for instance, the model of a star radiating in the surrounding atmosphere. It is assumed that the sphere, as well as the surrounding medium, are emitting with a Planckian distribution

$$B(T,\nu) = \frac{2h\nu^3}{c^2}\frac{1}{e^{\frac{h\nu}{k_bT}}-1} \tag{6.31}$$

with h and k_b being the Planck and Boltzmann constant, and c the speed of light.

To make the problem tractable, the authors of [98] neglect scattering entirely by setting $\sigma \equiv 0$. Moreover, the absorption coefficient is modeled by a constant $k(x,\nu) = k_\nu$, with ν being the frequency. The emission term is modeled by $f(x,\nu) = k_\nu B(T_m,\nu)$, resulting in the following form of the radiative transfer equation (6.12),

$$\frac{1}{c}\frac{\partial u}{\partial t} + \omega \cdot \nabla_x u = k_\nu (B(T_m,\nu) - u), \quad (t,x,n,\nu) \in D_T \times S \times \Lambda. \tag{6.32}$$

In the context of radiation-(magneto)hydrodynamics, one is mostly interested in the angular moments of the radiative intensity that naturally arise in calculating the contribution of radiation to the total energy of the fluid (plasma). Hence, it is customary to integrate (6.32) over the sphere S to derive the following PDE for incident radiation (6.16):

$$\frac{1}{c}\frac{\partial}{\partial t}G + \nabla_x \cdot F = k_\nu \Big(b(T_m,\nu) - G\Big), \quad (t,x,\nu) \in D_T \times \Lambda. \tag{6.33}$$

with $b(T,\nu) = 4\pi B(T,\nu)$.

However, the PDE (6.33) is not closed and one needs a closure for the flux F in terms of the incident radiation G. It is common practice in astrophysics to use the so-called *diffusion approximation* of the flux [100]:

$$F(t,x,\nu) = -\frac{1}{3k_\nu}\nabla G(t,x,\nu), \tag{6.34}$$

resulting in the following PDE,

$$\frac{1}{c}\frac{\partial}{\partial t}G - \frac{1}{3k_\nu}\Delta G = k_\nu \Big(b(T_m,\nu) - G\Big), \quad (t,x,\nu) \in D_T \times \Lambda. \tag{6.35}$$

Defining the *Knudsen number* $K = Lk_\nu$ (with L being a characteristic length scale), it is well known that the diffusion approximation is justified in the limit of $K \to \infty$.

k_ν	N_{int}	N_{sb}	N_{tb}	$K-1$	$\bar{d}$	λ	$\mathcal{E}_T$	Training Time
1	16384	12288	12288	4	40	0.1	0.0028	3 hr 25 min
10	16384	12288	12288	4	40	0.1	0.012	2 hr 15 min

Table 6.5: Results for polychromatic time-dependent radiative transfer in three space dimensions.

Although the PDE (6.35) is simpler than the full radiative transfer equation (6.12), efficient numerical approximation of (6.35) is still quite challenging as the incident radiation is a function of 5 variables. As it happens, PINNs provide an efficient method for approximating high-dimensional parabolic equations such as (6.35), see [73] section 3 for details.

However, by assuming radial symmetry and with the flux approximation given in (6.34), the differential equation (6.35) admits an analytical solution satisfying the initial and boundary conditions

$$\begin{aligned} G(0,r,\nu) &= b(T_m,\nu), \\ G(t,r\to\infty,\nu) &= b(T_m,\nu), \\ G(0,R_i,\nu) &= b(T_s,\nu). \end{aligned} \tag{6.36}$$

The exact solution for (6.35) then reads [98],

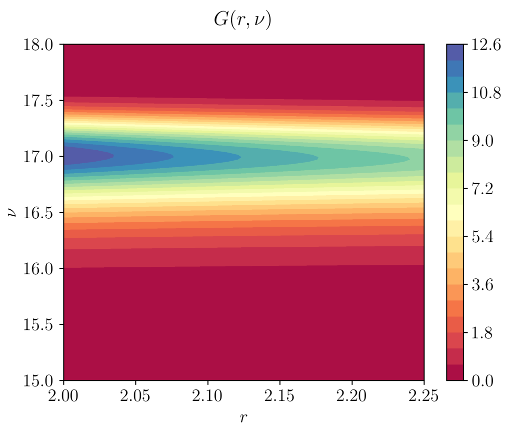
$$
\begin{aligned}
G(t,r,\nu) &= b(T_m,\nu) + \frac{R_i}{r}\Big(b(T_s,\nu) - b(T_m,\nu)\Big)F(t,r,\nu), \\
F(t,r,\nu) &= \frac{1}{2}\exp\left(-3k_\nu(r-R)\right)\Bigg\{\mathrm{Erfc}\Bigg(\sqrt{\frac{3k_\nu}{4ct}}(r-R) - \sqrt{k_\nu ct}\Bigg) + \mathrm{Erfc}
\end{aligned}
\tag{6.37}
$$

$$
bigg(\sqrt{\frac{3k_\nu}{4ct}}(r-R) + \sqrt{k_\nu ct}\Bigg)\Bigg\}.
$$

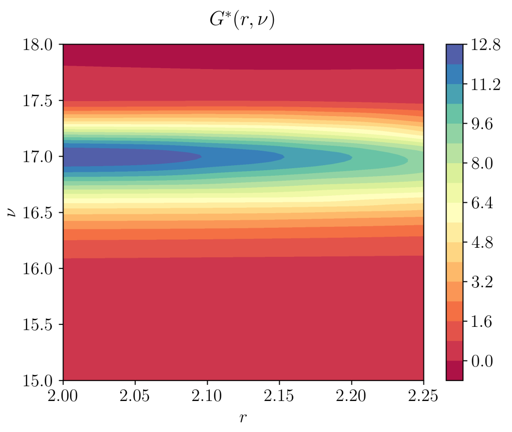
For this numerical experiment, we will approximate the full time-dependent radiative transfer equations (6.32) with the PINNs algorithm 1. To this end, we consider (6.32) in the spatial domain D enclosed between two spheres with radii $R_i = 2$ and $R_e = 4$. Moreover, we introduce an auxiliary temporal variable $\tau = ct$ to rescale time to $[0,1]$, whereas the energy group ν ranges between 10^{15} and 10^{18}. We set $T_s = 150eV$ and $T_m = 120eV$.

The PINNs algorithm 1 employs Sobol points in the interior, spatial and temporal boundary training sets and we set $N_{int} = 16384$, $N_{sb} = N_{tb} = 12228$. Moreover, we solve this problem for two different values of the (constant in frequency) absorption coefficient i.e $k_\nu = 1$ and $k_\nu = 10$, resulting in two different Knudsen numbers of $K = 2$ and $K = 20$, respectively. Given the challenging nature of this problem, we choose slightly different ranges of the hyperparameters, presented in tabel 6.1 for ensemble training and also use 20 retrainings, corresponding to different random starting values for the weights and biases in the training procedure. The resulting best performing configurations are reported in table 6.5. We observe from this table that PINNs provide a very low training error of 2.8×10^{-3}, for the $K = 2$ case. This training error is comparable to the training errors for the previous two examples. The training error increases by a factor of 4 for the $K = 20$ case, but still remains relatively low.

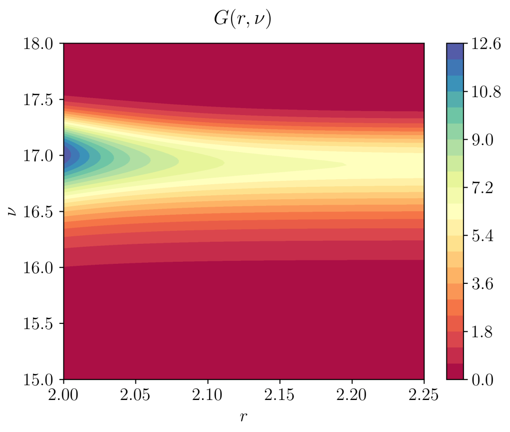
As we do not have exact analytical formulas for the full radiative intensity, it is not possible to compute generalization errors. However to ascertain the quality of the solution, we compare with the exact solution (6.37) of the diffusion equation (6.35) for the incident radiation. This comparison is shown as contour plots for the incident radiation in the (r,ν)-plane (with r denoting the radial direction) in figure 6.5 as well as one-dimensional cross-sections for different values of the radius r in figure 6.6. As seen from both these figures, there is good agreement between the incident radiation, computed by a Gauss quadrature of the PINN approximation to the radiative intensity in (6.32), and the analytical solution of the diffusion approximation (6.35) for the $K = 20$ case. This is not unexpected as the diffusion approximation is accurate for large Knudsen numbers. On the other hand, there is a significant difference between the the incident radiation, computed by a Gauss quadrature of the PINN approximation to the radiative intensity in (6.32), and the analytical solution of the diffusion approximation (6.35) for the $K = 2$ case. This follows from the fact that the diffusion approximation will provide a poor approximation of (6.32) for low Knudsen numbers. On the other hand, given the relatively low training error as well as the error estimate (9.3), coupled with the results of the previous numerical experiments, we argue that the PINN provides a much more accurate approximation to the underlying radiative intensity (and its moments) than the diffusion approximation will do, atleast for low to moderate Knudsen numbers. Hence, PINNs provide a viable and accurate method for competing radiative transfer in media with different optical properties. Moreover, the runtime for even this very complicated problem was reasonably small, ranging from two to three and half hours on a single GPU.



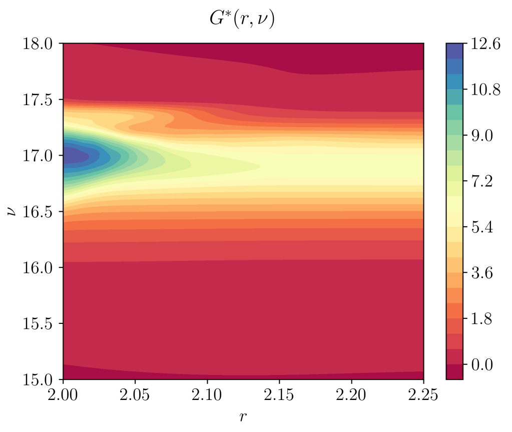
(a) Exact solution of (6.35) for $k_\nu = 1$



(b) PINN for $k_\nu = 1$



(c) Exact solution of (6.35) for $k_\nu = 10$



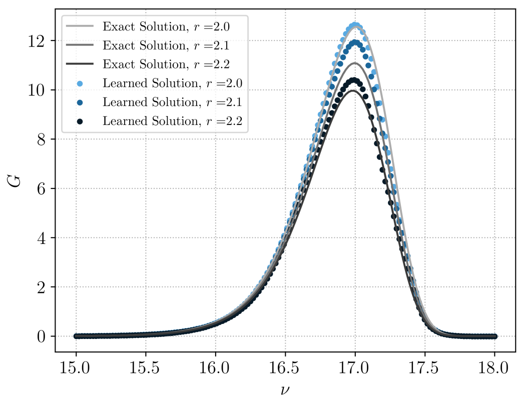
(d) PINN at $\tau = 1$ for $k_\nu = 10$

Figure 6.5: Comparison of incident radiation with respect to the exact solution of the diffusion approximation (6.35) and PINN approximation of the full radiative transfer equation (6.32) at rescaled time $\tau = 1$ for two different values of the absorption coefficient $k_\nu = 1, 10$

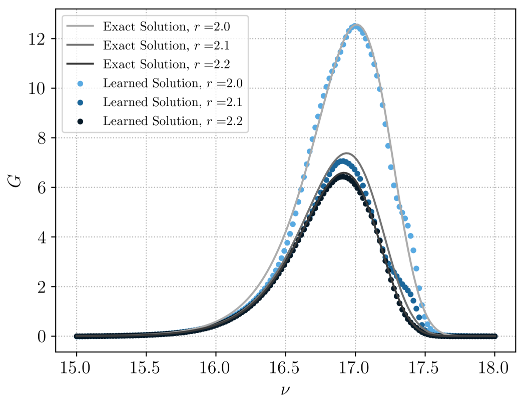
PINNs for the Inverse problem for radiative transfer

Next, we focus on a parameter identification problem (see Section 5.1) for radiative transfer. More specifically, we consider the monochromatic stationary version of the radiative transfer equation (6.12) in three space dimensions and choose the following concrete parameter identification problem:
Given measurements of the incident radiation $G(x)$, find the unknown absorption coefficient $k = k(x)$ and the resulting radiative intensity $u(x, \omega)$ which solves the stationary radiative transfer equation. The spatial domain is the unit cube $D = [0, 1]^3$, with scattering coefficient $\sigma = 0.5$, scattering kernel $\Phi \equiv 1$. The source term f and boundary term u_b are generated using the following synthetic absorption coefficient and exact solution,

$$k(x) = \prod_{i=1}^{3} x_i^2, \quad u(x, \omega) = \frac{3}{16\pi}(1 + (\omega \cdot \omega')^2) \prod_{i=1}^{3} x_i(x_i - 1), \quad n' = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)^T. \tag{6.38}$$



(a) Solution of (6.32) and (6.33) for $k_\nu = 1$



(b) Solution of (6.32) and (6.33) for $k_\nu = 10$

Figure 6.6: Comparison of exact solutions of the diffusion approximation (6.35) with the PINN approximation of the full radiative transfer equation (6.32) for two different values of the absorption coefficient $k_\nu = 1, 10$ at different radial locations and at rescaled time $\tau = 1$

N_{int}	N_{sb}	N_d	$K-1$	$\bar{d}$	λ	$\mathcal{E}_T$	$\|\|u-u^*\|\|_{L^2}$	$\|\|k-k^*\|\|_{L^2}$	$\|\|G-G^*\|\|_{L^2}$	Training Time
16384	120	4096	8	20	1.0	0.00094	0.65 %	2.8 %	0.073%	1 hr 44 min

Table 6.6: Results for the inverse problem for radiative transfer.

The measured incident radiation $\bar{G}$ in (6.40) is calculated from the radiative intensity u above by using the formula (6.16).

Following Section 5.1, we seek to find the deep neural networks $k_{\theta_k} : D \to \mathbb{R}_+$ and $u_{\theta_u} : D \times S \to \mathbb{R}$, with the concatenated parameter vector $\theta = \{\theta_k, \theta_u\} \in \Theta$, approximating the absorption coefficient and radiative intensity, respectively. The *interior* and *data residual* 5.4 are,

$$r_{int}[u_{\theta_u}, k_{\theta_k}] := u_{\theta_u} + \omega \cdot \nabla_x u_{\theta_u} + k_{\theta_k} u_{\theta_u} + \sigma \left(u_{\theta_u} - \frac{1}{s_d} \sum_{i=1}^{N_S} w_i^S u_{\theta_u}(x, \omega_i^S) \right) - f. \tag{6.39}$$

$$r_d[u_\theta] := G\left(u_{\theta_u}\right) - \bar{G}(x), \quad \forall x \in D, \tag{6.40}$$

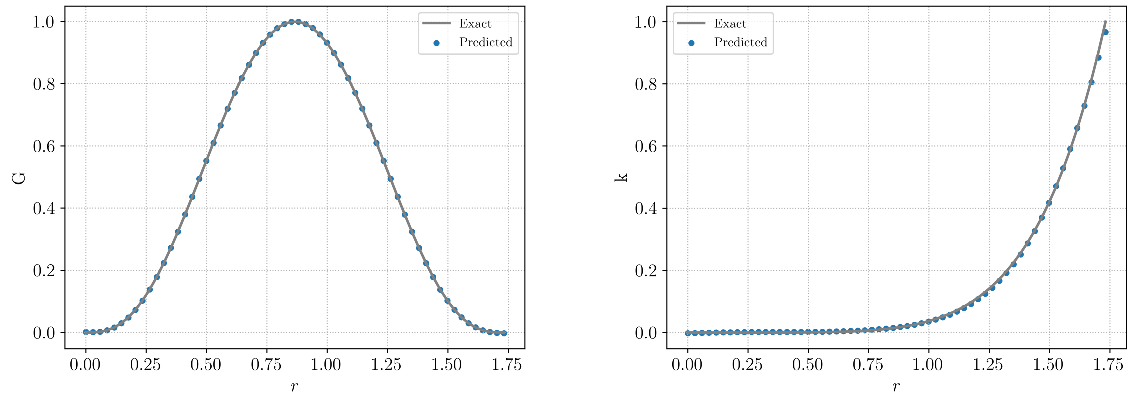
with G being the incident radiation calculated from (6.16) with a Gauss quadrature approximation of the angular integral and $\bar{G}$ being the measured incident radiation.

Clearly this inverse problem is ill-posed as multiple absorption coefficients might lead to the same incident radiation. Therefore, in order to ensure uniqueness of the absorption coefficient we also impose boundary conditions on the neural network approximating the absorption coefficient k_{θ_k} to approximately match the values of k, defined in (6.38) on the boundary of D and include in the loss function the so-called Tikhonov regularization:

$$J_T(\theta) = \lambda_k ||\nabla k_\theta||_2^2, \quad \lambda_k = 0.001. \tag{6.41}$$

We use Sobol points for the interior training points and uniformly distributed random points are used as data training points, with $N_{int} = 16384$, $N_d = 4096$. The resulting best performing hyperparameter configuration after ensemble training is presented in Table 6.6.

In figure 6.7 we plot the incident radiation G and the absorption coefficient k, along the diagonal of the unit cube. As observed from this figure, the incident radiation is almost identical to the measured data $\bar{G}$. This is further verified from table 6.6, from which we observe a very low L^2-error for the incident radiation. On the other hand, the absorption coefficient agrees reasonably well with the ground truth in (6.38), with an error of less than 3%. Also the radiative intensity is approximated to very high accuracy, with a generalization error below 1%. This is even more impressive if one consider that the problem is solved within a computational time of approximately 100 minutes.



(a) Incident radiation G (b) Absorption Coefficient k

Figure 6.7: PINNs results for the inverse problem for radiative transfer. The PINNs approximation to the incident radiation and absorption coefficient are plotted along the diagonal of the unit cube and compared with the measured data $\bar{G}$ and ground truth absorption coefficient k given by (6.38).

6.3 Kolmogorov Equations

Several different financial phenomena, including option pricing, are described by the high-dimensional Kolmogorov partial differential equation,

$$\begin{aligned} u_t &= \frac{1}{2}\mathrm{Tr}\left(\sigma(x)\sigma(x)^T H_x[u]\right) + \mu(x)^T \nabla_x u, && \forall (x,t) \in D \times [0,T] \\ u(x,0) &= \varphi(x), && \forall x \in D \\ u(y,t) &= \psi(y,t), && \forall (y,t) \in \partial D \times [0,T] \end{aligned} \tag{6.42}$$

In the financial context, the equation describes the evolution of an option price, $u : D \times [0,T] \to \mathbb{R}$, as a function of the portfolio assets' price $x \in D$. These might include, for instance, stocks and funds prices. Moreover, $\sigma : \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$ and $\mu : \mathbb{R}^d \to \mathbb{R}^d$ are functions describing is the assets volatility and the assets return, respectively. On the other hand, ∇_x and $H_x[u]$ are the gradient and the Hessian of u with respect to the coordinate x. In order to uniquely define a solution of the PDE, suitable boundary data $\psi(y,t)$, $\forall (y,t) \in \partial D \times [0,T]$ and initial condition $\varphi(x)$, $\forall x \in D$ have to be provided.

The main challenge in solving 6.42 is the high-dimensionality d of the *state space* $D \subset \mathbb{R}^d$, which might account hundreds or even thousands of different assets.

Prototypical examples of Kolmogorov PDEs include,

- *Heat Equation.* Let $\sigma = \sqrt{k}ID$, with k being the thermal diffusivity of the medium and $\mu = 0$. This results in the heat equation for the temperature u of the medium:

$$u_t = k\Delta_x u, \tag{6.43}$$

  with $\Delta_x = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}$.

- *Black-Scholes Equations.* Let σ, μ be linear functions, then the Kolmogorov equation 6.42 reduces to

$$u_t = \sum_{i,j=1}^{d} \beta_i\beta_j\rho_{ij}x_ix_ju_{x_ix_j} + \sum_{i=1}^{d}\mu_ix_iu_{x_i} \tag{6.44}$$

  where β_i and μ_i, $i = 1, .., d$, denote the volatility and interest rate of the stock prices x_i, $i = 1, .., d$, and $\rho_{i,j}$ the correlation between the stocks price. The initial condition ϕ can be interpreted as a payoff function. Common examples are the *basket call option* $\varphi(x) = \max\left(\sum_{i=1}^{d} a_ix_i - K, 0\right)$ and *call on max option* $\varphi(x) = \max\left(\max_i a_ix_i - K, 0\right)$

Our goal in this chapter is to approximate the classical solution of Kolmogorov equations with PINNs.

6.3.1 PINNs

As commonly done in the rest of thesis, we begin by describing the building elements of PINNs algorithm 1.

Training Sets

For definiteness, we set $D = [0, 1]^d$. Given the high-dimensionality of the partial differential equation ($d > 20$), we choose the training sets based on uniformly distributed random points. We then divide the training set into the following three sets, $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$

- Interior training points $\mathcal{S}_{int} = \{y_n\}$ for $1 \leq n \leq N_{int}$, with each $y_n = (x, t)_n \in D \times [0, T]$.
- Spatial boundary training points $\mathcal{S}_{sb} = \{z_n, \psi(z_n)\}$ for $1 \leq n \leq N_{sb}$ with each $z_n = (x_n, t_n)$ and each $x_n \in \partial D \times [0, T]$.
- Temporal boundary training points $\mathcal{S}_{tb} = \{x_n, \varphi(x_n)\}$, with $1 \leq n \leq N_{tb}$ and each $x_n \in D$.

Residuals

Next, we define the following residuals:

- Interior residuals:

$$r_{int}[u_\theta] := \partial_t u_\theta - \sum_{i,j=1}^{n} \beta_i\beta_j\rho_{ij}x_ix_j\partial_{x_ix_j}u_\theta - \sum_{i=1}^{n}\mu_ix_i\partial_{x_i}u_\theta, \quad \forall (x, t) \in D \times [0, T] \tag{6.45}$$

- Spatial boundary residuals,

$$r_{sb}[u_\theta] := u_\theta - \psi, \quad \forall (y,t) \in \partial D \times [0,T] \tag{6.46}$$

- Temporal Boundary residual

$$r_{tb}[u_\theta] := u_\theta - \varphi, \quad \forall x \in D \tag{6.47}$$

Loss Functions

Finally, the associated loss function to the problem 6.42 is the same as the one defined in (6.20), which we write below for sake of definiteness,

$$J(\theta) := \sum_{j=1}^{N_{sb}} w_j^{sb} |r_{sb}[u_\theta](z_j^{sb})|^2 + \sum_{j=1}^{N_{tb}} w_j^{tb} |r_{tb}[u_\theta](x_j^{tb})|^2 + \lambda \sum_{j=1}^{N_{int}} w_j^{int} |r_{int}[u_\theta](y_j^{int})|^2 \tag{6.48}$$

Estimate on the Generalization Error

The generalization error of PINN approximating the linear Kolmogorov equation, defined as

$$\mathcal{E}_G = \mathcal{E}_G(\theta^*) = ||u - u^*||_{L^2(D\times[0,T])} \tag{6.49}$$

can be bounded, similarly to to the previous chapters, by leveraging the stability of the PDE (see Theorem 4 in [88]), and by making use of suitable quadrature error bounds. Differently from before, given the random nature of the quadrature, only a probabilistic bound can be obtained. However, as proven in [88], such bound is independent of the dimensionality of the underlying PDE, demonstrating that PINNs can overcome the curse of dimensionality. Details and technicalities of the proof can be found in the paper [88].

6.3.2 Numerical Experiments

Heat equation in several space dimensions

The Forward Problem. As a first numerical example, we consider the linear heat equation 6.43 with $k = 1$ in the domain $[0,1]^d$ and for the time interval $[0,1]$, for different space dimensions d. We consider the initial data $\bar{u}(x) = \frac{\|x\|^2}{d}$. In this case, the explicit solution of the linear heat equation is given by

$$u(x,t) = \frac{\|x\|^2}{d} + 2t. \tag{6.50}$$

For different values of d ranging up to $d = 100$, we train PINNs with algorithm 1, by selecting randomly chosen training points, with respect to the underlying uniform distribution. Ensemble training, as outlined above, is performed in order to select the best performing hyperparameters among the ones listed in Table 3.1 and resulting errors are shown in 6.7. In particular, we present the relative percentage generalization error $\mathcal{E}_G$, readily computed from definition (6.49) and normalized with the L^2-norm of the exact solution (6.50). We see from the table that the generalization errors and training times are very low, less than 1%, achieved in 45 minutes for 10 spatial dimensions, and rise rather slowly (approximately linearly) with dimension, resulting in a low generalization error of 4.3%, even for 100 space dimensions. Note that we have not used *any explicit solution formulas*, such as the Feynman-Kac formulas, in algorithm 1. Still, PINNs were able to obtain low enough errors, comparable to supervised learning based neural networks that relied on the availability of an explicit solution formula [4, 5]. This experiment illustrates the ability of PINNs to overcome the *curse of dimensionality*, at least with random training points.

d	N_{int}	N_{sb}	N_{tb}	$K-1$	d	L^1**-reg.**	L^2**-reg.**	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$	Training Time
20	65536	65536	65536	4	20	0.0	10^{-6}	0.1	0.006	0.89%	27min
50	65536	65536	65536	4	20	0.0	10^{-6}	0.1	0.0056	2.6%	50min
100	65536	65536	65536	4	20	0.0	0.0	0.1	0.0035	4.3%	1h 45min

Table 6.7: Best performing hyperparameters configurations for the forward problem of the high-dimensional heat equation, for different values of the dimensions d.

The Data Assimilation Problem. Next we consider the data assimilation problem for the heat equation in the same setting described above. In particular, following 5.2, we set the observation as $D'_T = [a, 1-a] \times [a, 1-a]^d$ with $a = 0.4$ and the data term (5.10) is defined by restricting the exact solution (6.50) to this observation domain.

All the training sets $\mathcal{S}_{int}, \mathcal{S}_{sb}$ and $\mathcal{S}_d$ consists of points, chosen randomly and independently with the underlying uniform distribution. For all dimensions d considered here, we let $N_{int} = 8192$, $N_d = 6144$ and $N_{sb} = 2048$, resulting in a total of $N = N_{int} + N_{sb} + N_d = 16384$ training points.

On these training sets, the algorithm 1 is run with loss function (5.54) and the best performing hyperparameters are identified after ensemble training and presented in Table 6.8. The resulting L^2-error $\mathcal{E}_G = \|u - u^*\|_{L^2(D_T)}$ (in relative percentages and computed on a test set of 10^5 randomly chosen samples) is shown in Table 6.8. We observe from this table that the PINN error is very small and increases apparently linearly, with still very low errors of 2% for $d = 100$ space dimensions. These results are *striking* on account of the following factors,

- The linear increase of error with respect to dimension appears to overcome the well-known *curse of dimensionality*, even in the case of the inverse problem.
- The relative size of the observation domain D'_T with respect to the whole domain D_T, shrinks exponentially with dimension. Yet, the PINNs algorithm is able to reconstruct the entire solution field with high accuracy, from observations in this very small domain.
- We observe that the PINNs are able to approximate the very high-dimensional inverse problem for the heat equation with *greater accuracy for even smaller computational cost*, than for the forward problem. This surprising observation merits further investigation and highlights the potential of PINNs in solving inverse problems.

d	N_{int}	N_{sb}	N_d	$K-1$	$\bar{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$	Training Time
20	8192	6144	2048	4	20	10^{-6}	0.001	0.002	0.62%	13min
50	8192	6144	2048	4	20	10^{-6}	0.001	0.003	1.18%	39min
100	8192	6144	2048	4	20	10^{-6}	0.001	0.0023	2.6%	1h 11min

Table 6.8: Best performing hyperparameters configurations for the data assimilation problem of the high-dimensional heat equation, for different values of the dimensions d.

Black-Scholes with Uncorrelated Noise

Next, we consider the multidimensional Black-Scholes equation 6.44 in the domain $D_T = [0,T] \times [90,110]^d$ with uncorrelated noise ($\rho_{ij} = 0$, for all $i,j = 1,...,d$, $i \neq j$). We assume $\mu_i = \mu = -0.05$, for all $i = 1,...,d$ and $\beta_i = 0.1 + 0.02i$, $i = 1\ldots,d$. We further consider the *basket call option* initial condition $\varphi(x) = \max(\max_{i\in\{1,...,d\}} x_i - 100, 0)$.

Given this configuration, we proceed as for the heat equation and train a PINN for different values of d, up to $d = 200$, on randomly selected training points. The results are summarized in Table 6.9. The table shows that the generalization error of the PINN increases only linearly (similar to the training time) for progressively greater input dimension d. In particular, the error does not grow beyond 2.5% even for $d = 200$, reinforcing again the claim that PINNs overcome the curse of dimensionality.

d	N_{int}	N_{sb}	N_{tb}	$K-1$	d	λ	L^2-**reg.**	$\mathcal{E}_T$	$\mathcal{E}_G^r$	Training Time
1	32768	16384	16384	4	28	0.1	0	0.0052	0.74%	3m 30s
2	32768	16384	16384	4	24	0.1	0	0.0767	1.0%	4m 46s
10	32768	16384	16384	4	28	0.1	10^{-6}	0.0467	1.55%	11m 13s
20	32768	16384	16384	4	32	1	10^{-6}	0.0254	1.51%	18m 23s
50	32768	16384	16384	4	28	1	0	0.0097	1.62%	40m 59s
100	32768	16384	16384	4	28	0.1	0	0.0045	1.51%	1h 18m 28s
200	32768	16384	16384	4	32	0.1	0	0.0015	2.28%	2h 28m 55s

Table 6.9: Best performing hyperparameters configurations for the multi-dimensional Black-Scholes equation with uncorrelated noise, for different values of the dimensions d.

Black-Scholes with Correlated Noise

As a last numerical experiment, we consider again the Black-Scholes equation 6.44 in the domain $D_T = [0,T] \times [90,110]^d$, for d ranging from 2 to 200. However, this time we assume the model with correlated noise. In particular $\rho_{ij} = \langle \zeta_i, \zeta_j \rangle$, with $\zeta_k = (\Sigma_{k,1},\ldots,\Sigma_{k,d}) \in \mathbb{R}^d$ and $\Sigma = (\Sigma_{i,j})_{i,j=1}^d \in \mathbb{R}^{d\times d}$ being the lower triangular Cholesky factor of the matrix $Q \in \mathbb{R}^{d\times d}$ given by $Q_{ij} = 1$ for $i = j$ and $Q_{ij} = 0.5$ for $i \neq j$. We assume $\mu_i = \mu = -0.05$, for all $i = 1,...,d$ and $\beta_i = 0.1 + 0.02i$, $i = 1\ldots,d$, and consider the initial condition $\varphi(x) = \max(100 - \min_{i\in\{1,...,d\}}(x_i), 0)$.

An important issue to address in this case is the expensive computation of the double sum in the corresponding model (6.44). More precisely, we define a scalar ν to be

$$\nu := \sum_{i=1}^{d}\sum_{j=1}^{d} Z_{ij}, \quad Z_{ij} = \beta_i\beta_j\langle\zeta_i,\zeta_j\rangle_{\mathbb{R}^d}(\partial^2_{x_ix_j}u)(t,x), \quad Z_{ij} = Z_{ji}\ \forall i,j. \tag{6.51}$$

The matrix $Z = (Z_{ij})_{ij} \in \mathbb{R}^{d\times d}$ is symmetric and hence only the upper triangular part, including the diagonal is required to compute the scalar ν. Thus, equation (6.51) can be rewritten as

$$\nu = \sum_{i=1}^{d}\sum_{j=i+1}^{d} 2Z_{ij} + \sum_{i=1}^{d} Z_{ii} \tag{6.52}$$

For $d \in \mathbb{N}$, computing ν in the form of (6.51) requires computing the d^2 elements of Z, however one needs to evaluate only $\frac{d}{2}(d+1)$ entries of Z by adapting equation (6.52) and hence saving $\frac{d}{2}(d-1)$ computations. For large d, that makes a big difference in training PINNs (computational time-wise) since ν has to be computed in each training iteration.

In Table 6.10 we report the results obtained by training PINNs on a set of randomly chosen training samples. The results are consistent with what we obtained in the previous numerical experiments. In fact, in this case, they are even more impressive, as the generalization error remain almost constant with increasing number of dimensions.

d	N_{int}	N_{sb}	N_{tb}	$K-1$	d	λ	L^2-**reg.**	$\mathcal{E}_T$	$\mathcal{E}_G^r$	Training Time
2	32768	16384	16384	4	24	1	0	0.0040	0.14%	3m 34s
10	32768	16384	16384	4	28	0.1	10^{-6}	0.0277	1.11%	8m 32s
20	32768	16384	16384	4	24	1	0	0.0198	1.18%	20m 03s
50	32768	16384	16384	4	28	1	0	0.0099	0.69%	38m 03s
100	32768	16384	16384	4	28	0.1	0	0.0064	0.77%	1h 53m 23s

Table 6.10: Best performing hyperparameters configurations for the multi-dimensional Black-Scholes equation with correlated noise, for different values of the dimensions d.

7 Operator Learning

So far we have only considered the scenario where we aim at directly solving a partial differential equation with no or little data. In this context, PINNs represent a very appealing learning framework, as it has been proven theoretically and numerically in the previous chapters. However, PINNs come with several drawbacks, in particular when it comes to training. An alternative to PINNs is using supervised learning, based on data generated by solving the underlying PDEs with standard numerical methods. This approach is particularly advantageous for many-query problems, like uncertainty quantification, Bayesian inversion, etc., where the underlying PDE is low dimensional in the state space and high-dimensional in the parameter space.

As a motivating example, let us consider the parameterized heat equation in the space-time domain $D_T = [0, L] \times [0, T]$, and parameter space $P = [0, 1]^n$ (with n even),

$$u_t(x, t, \mu) = u_{xx}(x, t, \mu), \quad x \in [0, L], t \in [0, T], \mu \in [0, 1]^n \tag{7.1}$$

Further assume that the parametric nature of the PDE stems from the initial condition

$$\bar{u}(x, \mu) = \sum_{i=1}^{\frac{n}{2}} \mu_i \sin\left(2i\pi \frac{x}{L}\right) + \sum_{i=\frac{n}{2}}^{n} \mu_i \cos\left(2i\pi \frac{x}{L}\right) \tag{7.2}$$

with $\mu \sim \mathrm{Unif}\left([0, 1]^n\right)$, drawn from a uniform distribution. We are interested in the solution of the PDE $u(x, T, \mu)$ at a later time $t = T$. We addressed already a similar example in Section 6.1. Clearly, the equation can be readily and efficiently solved with standard numerical methods for a single realization of the parameter μ. However, in order to solve many-query problems the equation has to be solved several times, making these problems extremely expensive. A natural choice to reduce the computational cost of many-query problem is to approximate the underlying map

$$(x, \mu) \mapsto u(x, T, \mu) \tag{7.3}$$

with neural networks:

$$u(x, T, \mu) \approx u_\theta(x, T, \mu), \tag{7.4}$$

trained with data $\mathcal{S} = \{(x_i, \mu_j, u(x_i, T, \mu_j))\}$, $i = 1, ..., N, j = 1, ..., M$ generated by solving equation 7.1 with standard numerical methods, on a grid with spatial resolution $\Delta x := x_{i+1} - x_i$, $\forall i = 1, \ldots, N$, for M realizations $\mu_j \sim \mathrm{Unif}\left([0, 1]^n\right)$, $j = 1, \ldots, M$.

The task at hand may also be interpreted as an *operator learning task*, since, as a matter of fact, we aim at approximating a *function to function* map:

$$\bar{u}(x) \mapsto u(x, T) = \mathcal{G}(\bar{u})(x) \tag{7.5}$$

Formally, let $\mathcal{X} = \mathcal{X}(D, \mathbb{R}^{n_\mathcal{X}})$ and $\mathcal{Y} = \mathcal{Y}(D, \mathbb{R}^{n_\mathcal{Y}})$ be two separable, Banach spaces, with $D \subset \mathbb{R}^d$ being a bounded domain and $n_\mathcal{X}, n_\mathcal{Y} \in \mathbb{N}$. We refer to the space $\mathcal{X}$ as the input space (e.g. space of initial conditions) and the space $\mathcal{Y}$ as the output space (e.g solution space of the PDE at time T). Let

$\mathcal{G} : \mathcal{X} \to \mathcal{Y}$ be an operator that represents the solution operator of a PDE of interest. The ultimate goal is to approximate the operator $\mathcal{G}$ from finite data of measurements of input and output function pairs $\{\bar{u}_j(x), u_j(x)\}_{j=1}^M$. However, one usually only gets access to the point-wise evaluations of the functions $\bar{u}_j$ and u_j at prescribed points, $\{\bar{u}_j(x_i)\}_{i=1}^N$, $\{u_j(x_i)\}_{i=1}^N$. One naive approach to constructing a surrogate model of the underlying operator $\mathcal{G}$ is to use a straightforward feed-forward neural networks, in the form of 2.24, with N features and N labels corresponding to the discretized input and output functions. Let us denote the approximate operator $\mathcal{G}^* \approx \mathcal{G}$, then

$$\mathcal{G}^* : \mathbb{R}^N \to \mathbb{R}^N, \quad \mathcal{G}^* \left(\mathcal{E}_\mathcal{X}(\bar{u})\right) = \mathcal{E}_\mathcal{Y}(u), \tag{7.6}$$

with

$$\begin{aligned} \mathcal{E}_\mathcal{X} : \mathcal{X} \to \mathbb{R}^N, \quad \mathcal{E}_\mathcal{X}(\bar{u}) = (\bar{u}(x_1), \dots, \bar{u}(x_N)) \in \mathbb{R}^N, \\ \mathcal{E}_\mathcal{Y} : \mathcal{Y} \to \mathbb{R}^N, \quad \mathcal{E}_\mathcal{Y}(u) = (u(x_1), \dots, u(x_N)) \in \mathbb{R}^N, \end{aligned} \tag{7.7}$$

being *encoding operators*, which in this case represent the evaluation of the input and output functions at given sensor points. For instance, in the case of 7.2, $\mathcal{E}_\mathcal{X}(\bar{u}) = (\mu_1, ..., \mu_n)$, with μ_j being the coefficients of the truncated discrete Fourier transform of $\bar{u}$. However, the models described in Equation 7.6 with the encoding (7.7) do not serve as suitable operator learning surrogates, since it may not generalize well to different discretization, beside the one of the training data. Indeed, despite a rigorous definition of an operator learning model remains an active area of research, all existing definitions concur on a crucial requirement: *operator learning models must exhibit invariance with respect to the discretization of the input and output functions*. An initial attempt to formally define operator learning models was presented in [101]. The authors introduced a precise mathematical notion of discretization invariance and established three requirements for a model to qualify as a discretization-invariant model, referred to as *Neural Operators*:

1. acts on any discretization of the input function, i.e. accepts any set of points in the input domain,
2. can be evaluated at any point of the output domain,
3. converges to a continuum operator as the discretization is refined.

However, the proposed definition establishes the equivalence between the continuous and discrete models only in the infinite limit. Instead, more recently the authors of [102] proposed an alternative definition based on frames, according to which such *continuous-discrete equivalence* should hold at *any* level of discretization, not just in the limit. In contrast, the model described in (7.6), with the encoding operator given by the point-wise evaluation of the input and output functions, does *not* qualify as a neural operator according to both [102] and [101]. On the other hand, when the encoding is defined with respect to the the coefficient of the Fourier basis of the input-output function, the model represents a neural operator in the sense of [102]. Such a model is also know in the literature as Spectral Neural Operator [39].

Below we describe two popular classes of operator learning models, *DeepONet* and *Fourier Neural Operator*. In Section 7.3 we proposed a new learning architecture, named Neural Inverse Operator, which combines DeepONet and FNO, to approximate a large class of inverse problems for partial differential equations.

7.1 DeepONet

DeepONet has been first introduced in [103]. The architecture belongs to the class of models known as *operator networks*. Let $D \subset \mathbb{R}^{d_x}, U \subset \mathbb{R}^{d_u}$, and $\mathcal{X} = \mathcal{X}(D, \mathbb{R}^{n_\mathcal{X}})$ and $\mathcal{Y} = \mathcal{Y}(U, \mathbb{R}^{n_\mathcal{Y}})$ be suitable function

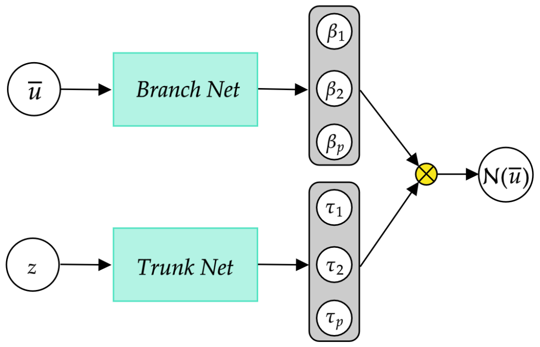


Figure 7.1: Schematic representation of DeepONet architecture

spaces. Then, a *DeepONet* is the operator, $\mathcal{N}^{DON} : \mathcal{X} \to \mathcal{Y}$, given by

$$\mathcal{N}^{DON}(\bar{u})(y) = \sum_{k=1}^{p} \beta_k(\bar{u})\tau_k(y), \quad \bar{u} \in \mathcal{X},\ y \in U, \tag{7.8}$$

where the *branch-net* β is a neural network that maps $\mathcal{E}(\bar{u}) = (\bar{u}(x_1), \dots, \bar{u}(x_m)) \in \mathbb{R}^m$, evaluations of the input $\bar{u}$ at sensor points $x := (x_1, \dots, x_m) \in D$, to $\mathbb{R}^p$:

$$\beta : \mathbb{R}^m \to \mathbb{R}^p,\ \mathcal{E}(\bar{u}) \mapsto (\beta_1(\mathcal{E}(\bar{u})), \dots, \beta_p(\mathcal{E}(\bar{u})), \tag{7.9}$$

and the *trunk-net* $\tau(y) = (\tau_1(y), \dots, \tau_p(y))$ is another neural network mapping,

$$\tau : U \to \mathbb{R}^p, \quad y \mapsto (\tau_1(y), \dots, \tau_p(y)). \tag{7.10}$$

Thus, a DeepONet combines the branch net (as coefficient functions) and trunk net (as basis functions) to create a mapping between functions. DeepONet, as defined above, does not represent a neural operator. In fact, although the trunk-net can be queried at any point, and thus, the output function can be obtained at any point, the branch-net constraints the input to fixed sensor locations. A way to circumvent this limitation is by interpolation of the input function, or by using the PCA-based approach proposed in [104, 105].

7.2 Neural Operators

As proposed in [101], a Neural Operator $\mathcal{N}^{NO}$ is formulated as iterative architecture

$$\mathcal{N}^{NO}(\bar{u})(x) = Q \circ \mathcal{L}_T \circ \dots \mathcal{L}_t \circ \dots \circ \mathcal{L}_1 \circ R(\bar{u})(x) \tag{7.11}$$

It consists of three main building pieces:

1. *Lifting.* The lifting operator R is defined as a linear or non linear transformation:

$$R : \mathcal{X}(D, \mathbb{R}^{n_\mathcal{X}}) \to \mathcal{Z}(D, \mathbb{R}^{d_v}), \quad R : \bar{u} \mapsto v_1$$

   with $d_v > n_\mathcal{X}$ and $\mathcal{Z}(D, \mathbb{R}^{n_\mathcal{Y}})$, and being a suitable Banach space. In other words, the lifting operator transforms (or lifts) the input to an high-dimensional *latent space*. Usually R is parameterized with a fully-connected neural network, or a simple linear layer.

2. *Iterative Kernel Integration.* The kernel operator is usually defined as follows:

$$k_t = \mathcal{K}(v_t)(x) = \int_D \kappa^{(t)}(x,y) v_t dy, \quad \forall x \in D \tag{7.12}$$

with $\kappa^{(t)} \in C(D \times D; \mathbb{R}^{d_v} \times \mathbb{R}^{d_v})$ being a *kernel function.* Then the hidden layers $\mathcal{L}_t$, $t = 1, ..., T$ are defined as

$$v_{t+1}(x) = \mathcal{L}_t(v_t)(x) = \sigma\left(W_t v_t(x) + b_t(x) + \mathcal{K}(v_t)(x)\right), \tag{7.13}$$

where $W_t \in \mathbb{R}^{d_v} \times \mathbb{R}^{d_v}$ is a *local linear operators* and $b : D \to \mathbb{R}^{d_v}$ is a *bias function.*

3. *Projection.* The final operator Q consist of a projection to the output function space:

$$Q : \mathcal{Z}(D, \mathbb{R}^{d_v}) \to \mathcal{Y}(D, \mathbb{R}^{n_\mathcal{Y}}), \quad Q : \bar{v_L} \mapsto u$$

with $d_v > n_\mathcal{Y}$.

The crucial difference between a Neural Operator as proposed in [101] and a standard feed-forward neural network is that all operations are directly defined in function space and therefore do not depend on any discretization of the data. However, the implementation of an iterative architecture as defined in (7.11) comes with a very high computational cost associated with the computation of the integral term. Let $\{x_1, x_2, ..., x_N\} \subset D$ be a set of N quadrature points ($N = N_1 \cdot N_2 \cdot ... \cdot N_{d_x}$, in the case of Cartesian grid-points), with associated weights $\{w_1, w_2, ..., w_N\}$, then

$$k_t(x_n) = \sum_{i=1}^{N} w_i \kappa(x_n, x_i) v_t(x_i), \quad n = 1, \ldots, N. \tag{7.14}$$

Hence, to compute k_t on the entire set of pints requires $\mathcal{O}(N^2)$ matrix-vector multiplications.

Several architectures, including Graph Neural Operator (GNO), Low-rank Neural Operator (LNO), Fourier Neural Operator (FNO), etc., have been introduced in order to efficiently compute the kernel integral and alleviate the computation cost of implementing Neural Operators [101].

7.2.1 Fourier Neural Operators

To mitigate the computational cost associated with computing the integral kernel, the authors of [35] proposed to represent and parameterize the kernel function directly in the Fourier space, leveraging the Fast Fourier Transform (FFT) algorithm with nearly linear complexity. The resulting architecture is termed *Fourier Neural Operator* (FNO).

In particular, by letting $\kappa^{(t)}(x,y) = \kappa^{(t)}(x-y)$ in (7.12) and exploiting the convolution theorem, the integral kernel can be explicitly formulated as:

$$K_t v_t = \mathcal{F}^{-1}\left[\mathcal{F}[\kappa](k) \cdot \mathcal{F}[v_t](k)\right]$$

where $\mathcal{F}[v]$ represents the Fourier transform of a function v, and $\mathcal{F}^{-1}[v]$ its inverse. In the discrete setting, the Fourier transform is replaced by the Fourier coefficients $\mathcal{F}_N v_t(k)$ of the discrete Fourier transform (DFT) of the function $v_t(x)$, with the corresponding modes (in each spatial direction) truncated up to k_{max}. These coefficients are computed based on an N_i-points grid in *each* spatial direction, $i = 1, ..., d_x$. Moreover, the DFT of κ at the frequency k is parameterized as complex-valued $(d_v \times d_v)$-tensor $P_t(k) \in \mathbb{C}^{d_v \times d_v}$. Then, the integral kernel simplifies to

$$K_t v_t = \mathcal{F}_N^{-1}\left[P_t(k) \cdot \mathcal{F}_N[v_t](k)\right], \quad k = 1, ..., k_{max}^{d_x}.$$

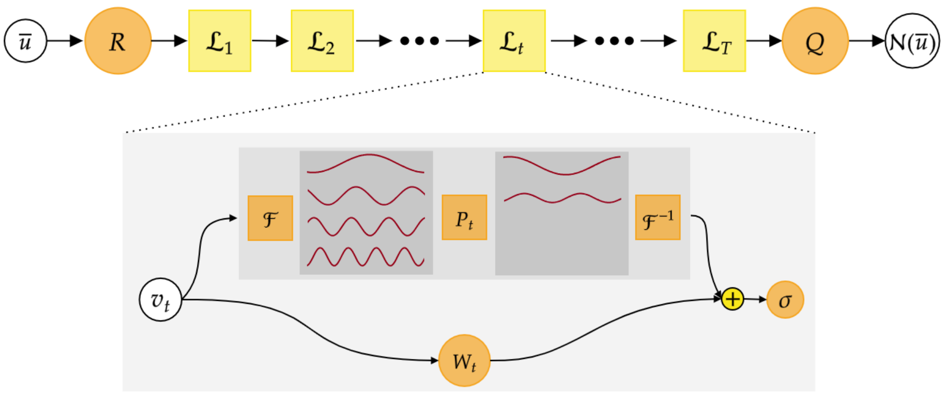


Figure 7.2: Schematic representation of Fourier Neural Operator architecture

See Figure 7.2 for a schematic representation of the architecture).

By utilizing the FFT, the computation complexity of the integral kernel reduces to $O(N \log N)$, which corresponds to the complexity of the FFT, while the total cost of the inner matrix-vector computation (of complexity $\mathcal{O}(k_{\max}^{d_x})$) is overall negligible given that k_{max} is usually small. However, this approach requires evaluating the input-output function pair on a Cartesian grid.

In the rest of the thesis, we assume that the lifting operation R is a linear function. For simplicity, we denote the composition of nonlinear operators $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_T$ followed by Q as $\mathcal{M}$. Therefore, the FNO neural operator $\mathcal{N}^{\text{FNO}}$ can be expressed as:

$$\mathcal{N}^{\text{FNO}}(\bar{u})(x) = \mathcal{M} \circ R(\bar{u})(x). \tag{7.15}$$

7.3 Neural Inverse Operators for solving PDE Inverse problems

In this section we focus on the class of inverse problems outlined in Section 2.3.1, which are only well-defined as mappings from operators to functions. This class of problems includes problems such as electrical impedance tomography, inverse wave scattering, optical imaging and seisimic imaging.

We begin by recalling the abstract formalism for the class of PDE inverse problems that we consider herein.

7.3.1 Mathematical Framework.

Let $D \subset \mathbb{R}^d$ be a bounded open set, with (smooth) boundary ∂D. Let $T > 0$ and $D_T = D$ or $D_T = D \times (0, T)$, depending on whether the PDE is time-(in)dependent. Correspondingly, $\partial D_T = \partial D$ or $\partial D_T = \partial D \times (0, T)$, respectively. Let $a \in \mathcal{A}(D)$, with $\mathcal{A}$ denoting a suitable function space over D, be a coefficient. Then, an abstract PDE can be written as

$$\mathcal{D}_a(u) = s, \quad \mathcal{B}(u) = g, \tag{7.16}$$

where $u \in \mathcal{U}(D_T)$ is the solution, $s \in \mathcal{S}(D_T)$ is the source term and $g \in \mathcal{B}(\partial D_T)$ is the boundary condition, for the PDE (7.16). Here, $\mathcal{D}_a : \mathcal{U} \to \mathcal{S}$ and $\mathcal{B} : \mathcal{U} \to \mathcal{B}$ are the differential and boundary operators, respectively.

The *forward problem* for the abstract PDE (7.16) amounts to the following: given the coefficient $a \in \mathcal{A}$, source term $s \in \mathcal{S}$ and boundary condition $g \in \mathcal{B}$, find the solution $u \in \mathcal{U}$ of the PDE (7.16).

However, in practice, one is often interested in the *inverse problem* associated with the PDE (7.16). For the specific class of problem we consider here, the inverse problem consists in reconstructing the coefficient a, given measurements of the solution u recorded at the boundary of the domain D_T. Without loss of generality, we denote such measurements as $\Psi \in \mathcal{H}(\partial\Omega)$. In general, a single instance (or small number) of boundary conditions g and measurements Ψ of the corresponding solutions u, do not suffice in inferring the underlying coefficient a. Instead, a collection of a all input-output pairs (g, Ψ) is needed in order to uniquely reconstruct the underlying coefficient.

Formally, the observable for this class of problem is a *boundary operator*:

$$\Lambda_a : \mathcal{B}(\partial D_T) \to \mathcal{H}(\partial D_T), \tag{7.17}$$

Then, the inverse map

$$\mathcal{F}^{-1} : \mathcal{L}\left(\mathcal{B}(\partial D_T), \mathcal{H}(\partial\Omega)\right) \to \mathcal{A}(D_T),\ \Lambda_a \mapsto a = \mathcal{F}^{-1}(\Lambda_a). \tag{7.18}$$

is unique and weakly stable ([106, 107, 108]).

Below, we provide four concrete examples of PDE inverse problems to which this abstract framework applies.

1. **Calderón Problem (Electrical Impedance Tomography).**

   Let us restrict ourselves to the case of time independent problem, $D_T = D$. Let the coefficient $0 < a \in C^2(D)$ represent the conductivity of the underlying medium (domain $D \subset \mathbb{R}^d$) and the associated PDE (7.16) is the following elliptic equation,

$$\begin{aligned} -\nabla \cdot \Big(a(z)\nabla u\Big) &= 0, \quad z \in D, \\ u(z) &= g(z), \quad z \in \partial D, \end{aligned} \tag{7.19}$$

   with Dirichlet boundary value $g \in H^{\frac{1}{2}}(\partial D)$ representing the voltage and the current source term is $s = 0$. The associated boundary observation operator Λ_a is the well-known *Dirichlet-to-Neumann* (DtN) map,

$$\begin{aligned} &\Lambda_a : H^{1/2}(\partial D) \to H^{-1/2}(\partial D), \\ &\Lambda_a[g] = a\frac{\partial u}{\partial \nu}\Big|_{\partial D}, \ \forall g \in H^{1/2}(\partial D), \end{aligned} \tag{7.20}$$

   which maps the input voltage g into the current $a(z)\frac{\partial u}{\partial \nu} = a\,\nabla u \cdot \nu$ (with ν being the unit outward normal vector) at the boundary and u is the solution of (7.19).

   The inverse problem, often referred to as the Calderón problem, constitutes the basis of EIT [45]. It aims to find the conductivity a of the medium, given different measurements of the DtN (voltage-to-current) pairs. Thus, this inverse problem falls into the considered abstract formalism and the inverse map (2.16) is given by,

$$\begin{aligned} &\mathcal{F}^{-1} : \mathcal{L}\left(H^{1/2}(\partial D), H^{-1/2}(\partial D)\right) \to C^2(D), \\ &\mathcal{F}^{-1} : \Lambda_a \mapsto a = \mathcal{F}^{-1}(\Lambda_a), \end{aligned} \tag{7.21}$$

with $\mathcal{L}(\cdot,\cdot)$ denoting the corresponding bounded linear operators. This inverse problem is shown to be well-defined and (logarithmic-) stable [106].

2. **Inverse Wave Scattering.**

   In many applications of interest, wave propagation in the frequency domain is used to infer material properties of the medium, modelled by the squared slowness $0 < a \in L^\infty(D)$. The associated PDE is the Helmholtz equation,

$$\begin{aligned} -\Delta u - \omega^2 a(z) u = 0, \quad z \in D, \\ u(z) = g(z), \quad z \in \partial D, \end{aligned} \tag{7.22}$$

   for some frequency ω and Dirichlet boundary condition $g \in H^{\frac{1}{2}}(\partial D)$. The resulting boundary observation operator is again the Dirichlet-to-Neumann (DtN) map

$$\begin{aligned} &\Lambda_a : H^{1/2}(\partial D) \to H^{-1/2}(\partial D), \\ &\Lambda_a[g] = \frac{\partial u}{\partial \nu}\Big|_{\partial D}, \ \forall g \in H^{1/2}(\partial D), \end{aligned} \tag{7.23}$$

   where u is the solution to (7.22) with the coefficient a. The corresponding inverse problem amounts to inferring the wave coefficient a from the DtN map (7.23). Thus, it can be formulated similar to the inverse map (7.21). Its well-posedness and stability have been demonstrated for the Helmholtz equation in [107] and references therein.

3. **Optical Imaging.**

   In optical imaging or tomography, the material properties of the medium $D \subset \mathbb{R}^d$ are expressed in terms of the scattering and absorption coefficients, $0 \le a, \sigma_a \in C(D)$. The associated PDE is the well-known radiative transport equation (RTE) for the particle density $u(z,v)$ at location $z \in D$ and velocity $v \in V \subset \mathbb{R}^d$, given by

$$\begin{aligned} v \cdot \nabla_z u(z,v) + \sigma_a(z) u(z,v) = \frac{1}{\epsilon} a(z) \mathcal{Q}[u], \quad z \in D, \\ u(z,v) = \phi(z,v), \quad z \in \Gamma_-, \end{aligned} \tag{7.24}$$

   where

$$\mathcal{Q}[u] = \int k(v,v') u(z,v') dv' - u(z,v)$$

   is the collision term, ϵ is the Knudsen number,

$$\Gamma_\pm = \{(z,v) \in \partial D \times V : \pm n_z \cdot v \ge 0\}$$

   are the inflow (outflow) boundaries and n_z is the unit outer normal vector at $z \in \partial D$. Thus, the input to this problem is provided by the particle density, $u_{\Gamma_-} \in L^1(\partial D \times V)$, prescribed on the inflow boundary. The associated boundary observation operator Λ_a defined in (2.14) is the so-called *Albedo* operator,

$$\Lambda_a : L^1(\Gamma_-) \to L^1(\Gamma_+), \ \Lambda_a : u\big|_{\Gamma_-} = \phi \mapsto u\big|_{\Gamma_+}, \tag{7.25}$$

   that maps the incident boundary values on Γ_- to the observed boundary values on the outflow boundary Γ_+.

   The corresponding inverse problem aims to infer the medium properties characterized by the scattering and absorption coefficients a, σ_a from the measurements of the Albedo operator. It leads to the following inverse map,

$$\begin{aligned} &\mathcal{F}^{-1} : \mathcal{L}\left(L^1(\Gamma_-), L^1(\Gamma_+)\right) \to C(D), \\ &\mathcal{F}^{-1} : \Lambda_a \mapsto a = \mathcal{F}^{-1}(\Lambda_a). \end{aligned} \tag{7.26}$$

The well-posedness and Lipschitz-stability of this inverse map were shown in [108].

4. **Seismic Imaging.**

   Seismic imaging is widely used in geophysics to infer and reconstruct sub-surface material properties for various applications such as CO_2 storage monitoring and seismic hazard assessment. Given a domain $D \subset \mathbb{R}^d$, we are interested in reconstructing the velocity coefficient $0 < a \in L^\infty(D)$ by sending in acoustic waves from the top boundary into the medium and measuring the response in the time domain. The associated PDE is the acoustic wave equation,

$$u_{tt}(z,t) + a^2(z)\Delta u = s(z,t), \ (z,t) \in D \times [0,T], \tag{7.27}$$

   with a time-dependent source term s. Here, u is the pressure variation. The wave equation is supplemented with zero initial conditions, i.e., $u(\cdot,0) = u_t(\cdot,0) = 0$ and suitable boundary conditions. In particular, *sources* are placed on a subset of the boundary and it is common to consider point sources $s(t,z) = g(t)\delta_S(z)$ with $g(t) \in L^2([0,T])$ and $\delta_S(z)$ being the Dirac measure concentrated on a set $S \subset \partial D$. These waves are transmitted, reflected, and refracted through the medium. Under certain assumptions (see Sec. 2.3 in [109]), the *effective* source s can be treated as in $L^2([0,T] \times D)$, which ensures the well-posedness of the PDE. The resulting signal is recorded at a set of receivers $\mathcal{R} \subset \partial D$ on the boundary that take continuous measurement in time $[0,T]$. The boundary observation operator (2.14) for this wave inverse problem is the *Source-to-Receiver* (StR) operator,

$$\begin{aligned} &\Lambda_a : L^2([0,T] \times D) \to L^2([0,T]; X_{\mathcal{R}}), \\ &\Lambda_a : s \mapsto u\big|_{[0,T]\times\mathcal{R}}, \end{aligned} \tag{7.28}$$

   where $X_{\mathcal{R}}$ is the metric space for the (discrete) set $\mathcal{R}$. The inverse problem that underpins seismic imaging is

$$\begin{aligned} \mathcal{F}^{-1}: &\quad \mathcal{L}\left(L^2([0,T] \times D), L^2([0,T]; X_{\mathcal{R}})\right) \to L^\infty(D), \\ \mathcal{F}^{-1}: &\quad \Lambda_a \mapsto a = \mathcal{F}^{-1}(\Lambda_a), \end{aligned} \tag{7.29}$$

   with Λ_a being the StR operator (7.28). Thus, seismic imaging aims to infer the subsurface spatial medium properties from spatial-temporal StR signals. This process is also termed as *migration*, or *Full waveform Inversion* (FWI) in the literature [110]. There have been studies on the well-posedness of the inverse problem for the wave equation (7.27) [111, 112, 113] although they do not directly apply to the setting considered here.

7.3.2 Learning Task and Challenges

Thus, the solution of the inverse problem (7.18) boils down to inferring (learning) the inverse map $\mathcal{F}^{-1}$ from relevant data. Given sufficient training data in the form of pairs $\left(\Lambda_a, \mathcal{F}^{-1}(\Lambda_a)\right)$ (or given the injectivity of the forward map, data in the form of pairs (Λ_a, a)), we aim to learn the inverse map $\mathcal{F}^{-1}$ and evaluate it on *test* (unseen) data. This task is very challenging on account of the following factors:

1. The inputs to the inverse map $\mathcal{F}^{-1}$ (2.16) are specified on the boundaries ∂D_T whereas the output is the coefficient a, defined in the interior of the underlying domain D. Thus, there is a mismatch in the domains of the inputs and outputs for the inverse map $\mathcal{F}^{-1}$.

2. The learning task requires us to learn *mappings from operators to functions* for $\mathcal{F}^{-1}$ defined in (2.16).

3. In general, the inverse map $\mathcal{F}^{-1}$ (2.16) may only be weakly stable, for instance, either in terms of small values of the Hölder exponent α in (2.17) or even only logarithmic-stable. In these cases, the learning task can be very sensitive to noises from the input, and additional regularization terms might be necessary.

7.3.3 A Motivating (Formal) Calculation.

We start by providing a heuristic motivation for our proposed architecture to learn the inverse map (2.16). To this end and for definiteness, we consider the inverse wave scattering problem for the Helmholtz equation (7.22), presented in section 2.3.1. Given the domain $D \subset \mathbb{R}^d$, we consider the following eigenvalue problem with Neumann boundary conditions,

$$\begin{aligned} -\Delta\varphi_k &= \lambda_k\varphi_k, \quad \forall z \in D. \\ \frac{\partial\varphi_k}{\partial\nu}|_{\partial D} &= 0, \quad \int\limits_D \varphi_k dz = 0. \end{aligned} \tag{7.30}$$

By standard PDE theory [114], there exist eigenvalues $0 \le \lambda_k \in \mathbb{R}$ for $k \in \mathbb{N}$, and the corresponding eigenfunctions $\{\varphi_k\}$, $k \in \mathbb{N}$, form an orthonormal basis for $L^2(D)$. We fix $K \in \mathbb{N}$ sufficiently large and without loss of generality, we assume $\omega = 1$ in the Helmholtz equation (7.22) to consider the following Dirichlet boundary value problems,

$$\begin{aligned} -\Delta u_k - a(z)u_k &= 0, \quad z \in D,\ 1 \le k \le K, \\ u(z) &= g_k(z), \quad z \in \partial D, \end{aligned} \tag{7.31}$$

where $g_k = \varphi_k\big|_{\partial D}$. Using (7.30) and (7.31), we can prove, the following formal *representation formula* for all $1 \le k \le K$,

$$\int\limits_D a u_k \varphi_k dz = \int\limits_D \lambda_k u_k \varphi_k dz - \int\limits_{\partial D} g_k \frac{\partial u_k}{\partial \nu} d\sigma(z). \tag{7.32}$$

Proof. Multiplying u_k (the solution of (7.31)) to Eqn (7.30) and integrating over space, we obtain,

$$\int\limits_D u_k \Delta\varphi_k dz + \lambda_k \int\limits_D \varphi_k dz = 0$$

Integrating by parts in the above equation and using the Gauss-Green formula yields,

$$-\int\limits_D \langle\nabla u_k, \nabla\varphi_k\rangle dz + \int\limits_{\partial D} u_k \underbrace{\frac{\partial\varphi_k}{\partial\nu}}_{=0} ds(z) + \lambda_k \int\limits_D u_k\varphi_k dz = 0. \tag{7.33}$$

Note that $\frac{\partial\varphi_k}{\partial\nu}\big|_{\partial D} = 0$ follows from the Neumann boundary conditions in (7.30).

Similarly, multiplying the solution φ_k of the Neumann problem (7.30) to the Eqn (7.31) and repeating the above integration parts yields,

$$-\int\limits_D \langle\nabla u_k, \nabla\varphi_k\rangle dz + \int\limits_{\partial D} g_k \frac{\partial u_k}{\partial\nu} ds(z) + \int\limits_D a(z) u_k\varphi_k dz = 0. \tag{7.34}$$

Formula (7.32) follows by subtracting (7.33) from (7.34). □

The formula (7.32) can be used to construct an approximation to the coefficient $a \in L^2(D)$ in the following manner. Writing $a \approx \sum\limits_{\ell=1}^{K} a_\ell \varphi_\ell$ (using the orthonormality of φ's) for K sufficiently large, we can evaluate the coefficients a_ℓ by solving the following Matrix equation for $A = \{a_\ell\}_{\ell=1}^{K}$,

$$\begin{aligned} CA = B,\ C_{k\ell} &= \int\limits_D u_k \varphi_k \varphi_l dx, \quad \forall k, l, \\ B_k &= \int\limits_D \lambda_k u_k \varphi_k dz + \int\limits_{\partial D} g_k \frac{\partial u_k}{\partial \nu} d\sigma(z), \forall k. \end{aligned} \tag{7.35}$$

Further setting $\Psi_k = \frac{\partial u_k}{\partial \nu}$, we observe that the formal approximation of the coefficient a relies on the following building blocks,

- *Basis Construction.* The operations $\mathcal{B}_k : z \mapsto (\varphi_k(z), \lambda_k)$, $1 \le k \le K$, that form a basis. Note that they are independent of the coefficient a.
- *PDE Solve.* The operation $\mathcal{E}_k : (g_k, \Psi_k) \mapsto \big(\{u_j^k\}_{j=1}^K, \int\limits_{\partial D} g_k \Psi_k d\sigma(z)\big)$ that amounts to (approximately) inferring the coefficients $\{u_j^k\}_{j=1}^K$ of the solution u_k of the Helmholtz equation (7.31), given the Dirichlet g_k and Neumann Ψ_k boundary values. A part of the right-hand side term B_k is also appended to this operation. Once the coefficients u_j^k are computed, the approximation u_k to the solution of (7.31) is readily computed in terms of the basis $\{\varphi_k\}$ by setting $u_k \approx \sum\limits_{j=1}^{K} u_j^k \varphi_j$.
- *Mode Mixing.* The previous two operations were restricted to individual modes, i.e., to each k, for $1 \le k \le K$. However, to construct the coefficients C_{kl} in (7.35), we need to mix different modes. One way to do so is through multiplication. We denote this operation by $\mathcal{M} : \big(\{\varphi_k\}_{k=1}^K, \{u_k\}_{k=1}^K\big) \mapsto \Big(\{u_k \varphi_k \varphi_\ell\}_{k,\ell=1}^K, \{\lambda_k u_k \varphi_k\}_{k=1}^K\Big)$.
- *Matrix Inversion.* In the final step, we need to build the Matrix C in (7.35) and (approximately) invert it. This operation can be summarized by $\mathcal{I} : \Big(\{u_k \varphi_k \varphi_\ell\}_{k,\ell=1}^K, \{\lambda_k u_k \varphi_k\}_{k=1}^K\Big) \mapsto \sum_{j=1}^K a_j \varphi_j$, with $A = \{a_j\}$ being the solution of (7.35).

7.3.4 The Architecture.

The formal approximation of the inverse map $\mathcal{F}^{-1}$ (2.16) for the Helmholtz equation by formulas (7.32)-(7.35) cannot be directly used in practice as one cannot solve the PDE (7.31) without knowing the coefficient a. However, the building blocks enumerated above motivate either an iterative fixed-point procedure or, in our case, a learning algorithm approximating $\mathcal{F}^{-1}$ from data. To this end, we observe that the basis construction $z \mapsto \varphi_k(z)$ amounts to a particular instantiation of a trunk-net (7.10) of a DeepONet. Similarly, the PDE solve map $\mathcal{E}_k : (g_k, \Psi_k) \mapsto \{u_j^k\}_{j=1}^K$ is a particular instance of the application of a branch-net (7.9) of a DeepONet. Moreover, they can be combined in a DeepONet (7.8) to approximate the solutions u_k of the PDE (7.31). However, a DeepONet (7.8) is linear in its trunk-net basis functions and thus cannot represent the nonlinear mode mixing operator $\mathcal{M}$. Instead, one can do so by passing the outputs of the DeepONet to the nonlinear layers of an FNO (7.15), which also performs the final inversion operator $\mathcal{I}$.

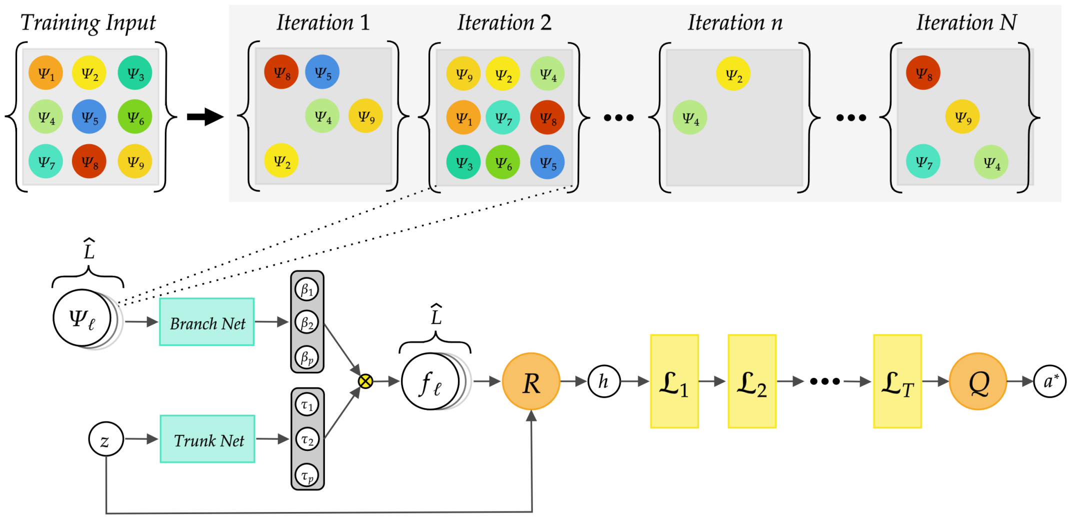


Figure 7.3: Schematic representation of Neural Inverse Operator (NIO) architecture, with R given by equation (7.38), and Randomized Batching.

These heuristic considerations are generalized to the abstract formalism of the inverse problem (2.16) and motivate us to propose the composition (stacking) of DeepONets and FNO to result in the following map,

$$\mathcal{N}_{NIO}: \begin{pmatrix} z \\ \{\Psi_\ell\}_{\ell=1}^L \end{pmatrix} \overset{\tau,\beta}{\longmapsto} \begin{pmatrix} \{\tau_k(z)\}_{k=1}^p \\ \{\beta_k\}_{k=1}^p \end{pmatrix} \overset{\mathcal{N}^{DON}}{\longmapsto} \{f_\ell(z)\}_{\ell=1}^L \overset{R}{\longmapsto} h(z) \overset{\mathcal{M}}{\longmapsto} a^*(z), \tag{7.36}$$

for approximating the abstract inverse map $\mathcal{F}^{-1}$ (2.16). Recall that $\mathcal{M}$ represents the composition of the non-linear operatos of FNO. Here, the linear map R can be explicitly defined as

$$h(z) = R(f_1, \dots, f_L, z) = \frac{1}{L}\sum_{\ell=1}^{L} D_\ell f_\ell + Ez, \tag{7.37}$$

with $E, D_\ell \in \mathbb{R}^{d_v}$. In other words, the inputs $z \in D$ and $\Psi_\ell = \Lambda_a(g_\ell)$ (2.14), for $1 \le \ell \le L$, are fed into the trunk- and branch-nets of a DeepONet $\mathcal{N}^{DON}$ (7.8), respectively, to create L representations $\{f_\ell\}_{\ell=1}^L$, defined in the interior of the underlying domain. These representations are first linearly transformed through R yielding $h(z)$, and finally *mixed* by the nonlinear component $\mathcal{M}$ of FNO resulting in an approximation of the underlying coefficient a^*. We observe that the DeepONet $\mathcal{N}^{DON}$ in $\mathcal{N}^{NIO}$ (7.36) is flexible enough to handle inputs defined on the boundary, i.e., Ψ_ℓ, and produce outputs f_ℓ, defined on the interior of the underlying domain.

It is important to note that the model takes only z and $\{\Psi_\ell\}_{\ell=1}^L$ as input, rather than z and the input-output pair $\{(g_\ell, \Psi_\ell)\}_{\ell=1}^L$. This choice is motivated by the following consideration: let us define μ_g as the underlying measure (distribution) on the boundary data g, where $g_\ell \sim \mu_g$, and $\mu_\Psi = \Lambda_a \# \mu_g$ represents the pushforward measure given the boundary operator Λ_a. Hence, $\Psi_\ell \sim \mu_\Psi$ for all $\ell = 1, ..., L$, with $\{\Psi_\ell\}_{\ell=1}^L$ representing the empirical distribution approximating μ_Ψ. This, together with the injectivity of the boundary observation operator (2.14), implies that $\{\Psi_\ell\}_{\ell=1}^L$ suffices to provide statistical information about the operator Λ_a given μ_g satisfying certain properties. We denote L as the number of samples discretizing the measure μ_Ψ or, with abuse of notation, the operator Λ_a.

The construction above addresses point (1) in Section 7.3. However, it does not necessarily satisfy point (2) in its current form. For the model to effectively process μ_Ψ as the input rather than a particular discrete realization of it, the following desirable properties should be met:

- The architecture should exhibit invariance under permutations of the input measurements $\{\Psi_\ell\}_{\ell=1}^L$ as they are i.i.d. samples of μ_Ψ.
- The learning framework must be able to handle an empirical measure of the distribution μ_Ψ with an arbitrary sample size $\tilde{L}$. In particular, the input sample size at the training and testing stages could be different.
- The performance of the model should be independent of the sample size $\tilde{L}$ used to discretize μ_Ψ.

To address the first two points, we can modify the architecture by redefining the linear transformation R as follows:

$$h(z) = R(f_1, \dots, f_L, z) = \frac{D}{L}\sum_{\ell=1}^{L} f_\ell + Ez, \quad E, D \in \mathbb{R}^{d_v} \tag{7.38}$$

To ensure the performance is independent of $\tilde{L}$, a naive and inefficient approach would involve constructing all possible permutations of $\{\Psi_\ell\}_{\ell=1}^L$ with size $\hat{L}$ for all $\hat{L} = 2, \dots, L$ and providing them as input to the model. However, inspired by the well-known *bagging* algorithm widely used in machine learning, we propose an efficient and novel method to tackle this point which we term *randomized batching.* In this approach, given the sequence $\{\Psi_\ell\}_{\ell=1}^L$, during each training iteration, an integer number $\hat{L}$ is randomly drawn from the set $\{2, \dots, L\}$. Then, $\hat{L}$ samples are randomly picked from $\{\Psi_\ell\}_{\ell=1}^L$ and the new sequence $\{\Psi_k\}_{k=1}^{\hat{L}}$ are fed into the model at each iteration during training.

We refer to the architecture (7.36) with the linear transformation R given in (7.38), and incorporating the *randomized batching* algorithm, as *Neural Inverse Operator*, or *NIO* in short (see Figure 7.3 for a schematic representation of the architecture).

7.4 Numerical Experiments.

We empirically test NIO on benchmark PDE inverse problems below. The exact details of the training, as well as the architecture and hyperparameter choices, are presented in the appendix, Sections 9.2.2 and 9.2.1.

As baselines in the following experiments, we choose two models. First, we consider a DeepONet (DONet) with a CNN as the branch net in (7.8), which performs mixing of the input function within the branch itself

$$\mathcal{N}_{DONet} : \begin{pmatrix} z \\ \{\Psi_\ell\}_{\ell=1}^L \end{pmatrix} \overset{\tau, \beta_{con}}{\longmapsto} \begin{pmatrix} \{\tau_k(z)\}_{k=1}^p \\ \{\beta_k\}_{k=1}^p \end{pmatrix} \overset{\mathcal{N}^{DON}}{\longmapsto} a^*(z). \tag{7.39}$$

Second, we consider a fully convolutional image-to-image neural network architecture (FCNN, details in the Appendix 9.2.1). A variant of this architecture was already used in seismic imaging (full waveform inversion) in [110]. We have extended this architecture significantly to apply it to the abstract inverse problem (2.16).

In all the experiments, the training (and test) data are generated by sampling from a probability distribution on the conductivity coefficient a. Once a sample conductivity is drawn, a set of Dirichlet boundary conditions $\{g_\ell\}_{\ell=1}^L$ are drawn from a probability distribution on the boundary values. For each g_ℓ, the

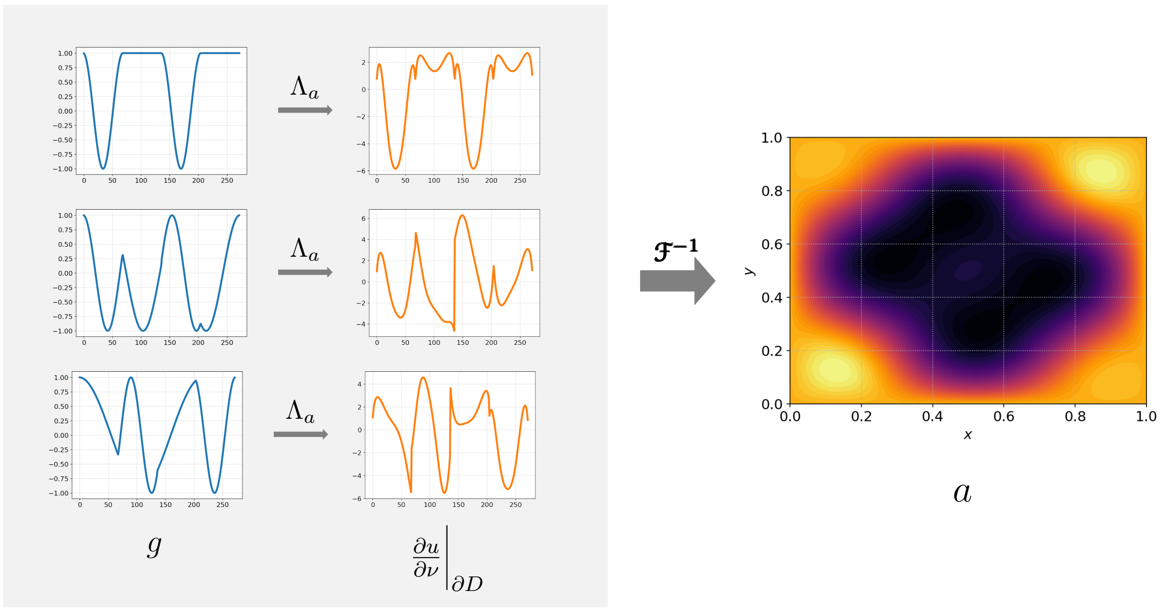


Figure 7.4: Illustration of a typical input (left) and output (right) sample for the Calderón Problem for EIT with trigonometric coefficients. The input is the Dirichlet-to-Neumann (DtN) map (7.20), represented here by three Dirichlet Boundary conditions (Voltage) to Current pairs, and the output is the conductivity coefficient a.

underlying elliptic equation is solved numerically and the corresponding observable, Ψ_ℓ, is evaluated on the boundary. Eventually the training set is defined as follows:

$$\mathcal{S}_{train} = \left\{ \left(\{\Psi_\ell^{(i)}\}_{\ell=1}^{L}, a^{(i)} \right),\ i = 1, ..., N \right\} \tag{7.40}$$

with N being the number of training samples. The total number of training samples used for different benchmarks are reported in Table 7.1.

7.4.1 Calderón Problem for EIT.

For the Calderón Problem for EIT we consider two different numerical examples.

1. We start with the Calderón problem for the elliptic equation (7.19) on the computational domain $D = [0, 1]^2$, with source $s = 0$. We choose the boundary data g_ℓ, for $1 \leq \ell \leq L = 20$ as the boundary values of

$$G_\ell(x, y) = \cos(\omega(x\cos(\theta_\ell) + y\sin(\theta_\ell))),$$

with $\theta_\ell = \frac{2\pi\ell}{20}$ and $\omega = 2\pi$. For the coefficient a, we sample from trigonometric functions by setting

$$a(x, y) = \exp\Big(\sum_{k=1}^{m} c_k \sin(k\pi x)\sin(k\pi y)\Big),$$

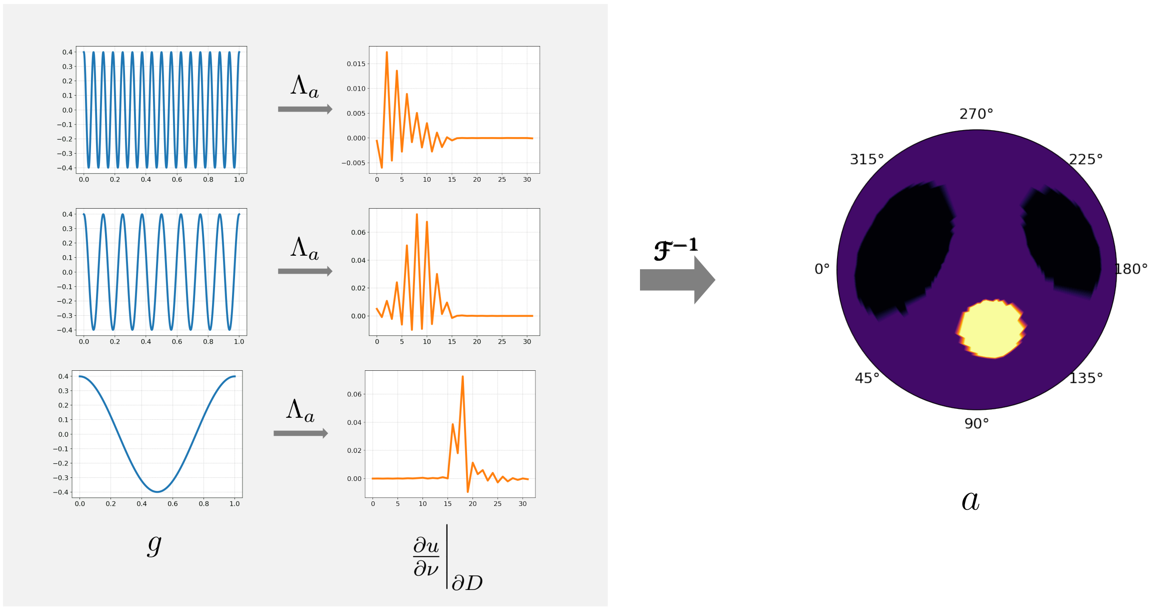


Figure 7.5: Illustration of EIT for the discontinuous heart-lung Phantom of [115]. Left: Input through the DtN (voltage-to-current) map. Right: Conductivity field showing the Phantom of the heart and lungs.

with $m = \text{Unif}(\{1, 2, 3, 4, 5\})$ and $\{c_k\} \sim \text{Unif}([-1, 1]^m)$.

2. As a second experiment for EIT, we consider a more practical example suggested in [115], where the authors model the EIT imaging of the heart and lungs of a patient using electrodes on the body. The underlying domain D is the unit circle and the boundary conditions are given by $g_\ell(\theta) = \frac{1}{2\pi}\exp(i2\pi\theta f_\ell)$, with $\ell = 1, \dots, 32$ and $f = [-16, \dots, -1, 1, 14, 15, 16]$. To describe the phantom for the heart and lungs, we define the following sets of points on the domain D:

$$s_h = \left\{(x, y) \in D \text{ s.t } \sqrt{e_{h,1}(x - c_{h,1})^2 + e_{h,2}(y - c_{h,2})^2} < 0.2\right\}$$

$$s_{l_1} = \left\{(x, y) \in D \text{ s.t } \sqrt{e_{l_1,1}\left(\cos(\alpha)x + \sin(\alpha)y - c_{l_1,1}\right)^2 + e_{l_1,2}\left(\cos(\alpha)y - \sin(\alpha)x - c_{l_1,2}\right)^2} < 0.5\right\}$$

$$s_{l_2} = \left\{(x, y) \in D \text{ s.t } \sqrt{e_{l_2,1}\left(\cos(\alpha)x + \sin(\alpha)y - c_{l_2,1}\right)^2 + e_{l_2,2}\left(\cos(\alpha)y - \sin(\alpha)x - c_{l_2,2}\right)^2} < 0.4\right\}$$

Here, $e_{h,1} = 0.8$, $e_{h,2} = 1$, $e_{l_1,1} = 3$, $e_{l_1,2} = 1$, $e_{l_2,1} = 3$, and $e_{l_2,2} = 1$ represent the eccentricities of the ellipses describing the heart and lungs. The center locations of the heart and lungs are given by $c_{h,1} = -0.1$, $c_{h,2} = 0.4$, $c_{l_1,1} = 0.5$, $c_{l_1,2} = 0.2$, $c_{l_2,1} = -0.6$, and $c_{l_2,2} = 0.1$ and the orientation of the lungs by $\alpha = \frac{\pi}{7}$. Then the body conductivity coefficient is defined as,

$$a(x, y) = \begin{cases} a_h & (x, y) \in s_h \\ a_{l_1} & (x, y) \in s_{l_1} \\ a_{l_2} & (x, y) \in s_{l_2} \\ a_b & \text{else} \end{cases}$$

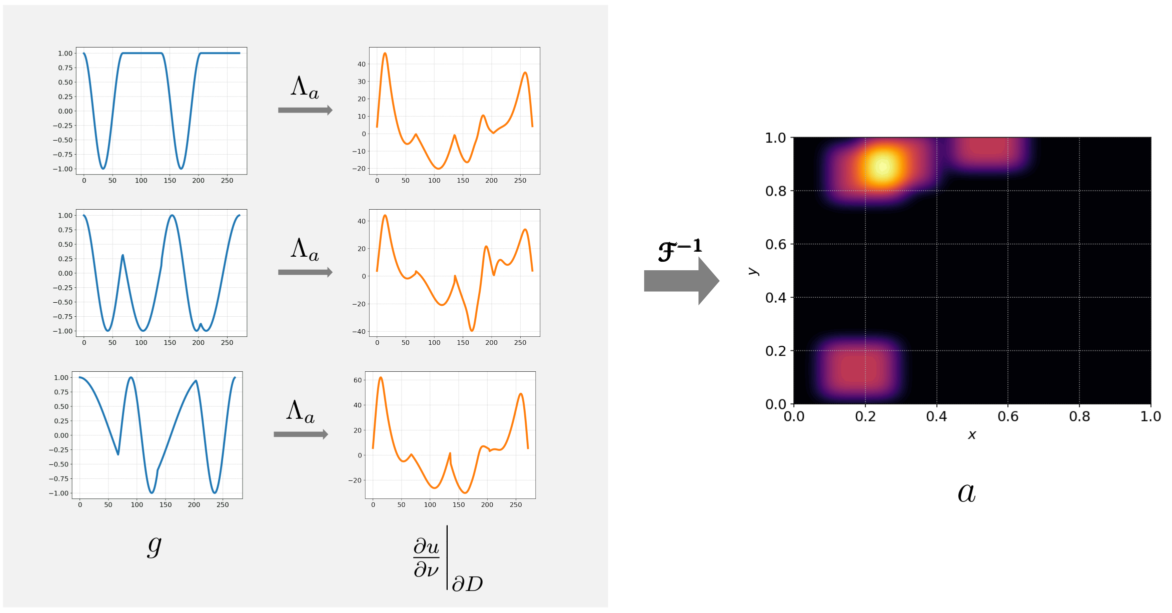


Figure 7.6: Illustration of detection of inclusions through the Inverse Wave Scattering with the Helmholtz equation. Left: Input represented through 3 samples for the DtN map. Right: Coefficient a.

with $a_h = 2$, $a_{l_1} = a_{l_2} = 0.7$, $a_b = 1$. The training coefficients are obtained by adding 8% white noise to all the parameters above. Specifically, given

$$p = [e_{h,1}, e_{h,2}, e_{l_1,1}, e_{l_1,2}, e_{l_2,1}, e_{l_2,2}, c_{h,1}, c_{h,2}, c_{l_1,1}, c_{l_1,2}, c_{l_2,1}, c_{l_2,2}, a_h, a_{l_1}, a_{l_2}],$$

we define the perturbed version of the parameter vector p as $\tilde{p} = p(1+0.08\xi)$, where $\xi \sim \mathcal{N}(0,1)$ is a random variable drawn from the standard normal distribution. The coefficient is then defined as:

$$\tilde{a}(x,y) = \begin{cases} \tilde{a}_h & (x,y) \in \tilde{s}_h \\ \tilde{a}_{l_1} & (x,y) \in \tilde{s}_{l_1} \\ \tilde{a}_{l_2} & (x,y) \in \tilde{s}_{l_2} \\ \tilde{a}_b & \text{else} \end{cases}$$

where $\tilde{s}_h$, $\tilde{s}_{l_1}$ and $\tilde{s}_{l_2}$ are defined as above, but with the parameters replaced by their corresponding perturbed values in $\tilde{p}$.

The input of the learning operators is then obtained by computing the Fourier transform at frequencies f of the difference between the Neumann trace of the PDE solution with the coefficient a and the one with the unit coefficient $a = 1$.

7.4.2 Inverse Wave Scattering.

In this problem, the Helmholtz equation (7.22) is considered on the domain $D = [0,1]^2$, and the task is to learn coefficients sampled from a distribution,

$$a(x,y) = \sum_{k=1}^{m} \exp\big(-c(x-c_{1,k})^4 - c(y-c_{2,k})^4\big),$$

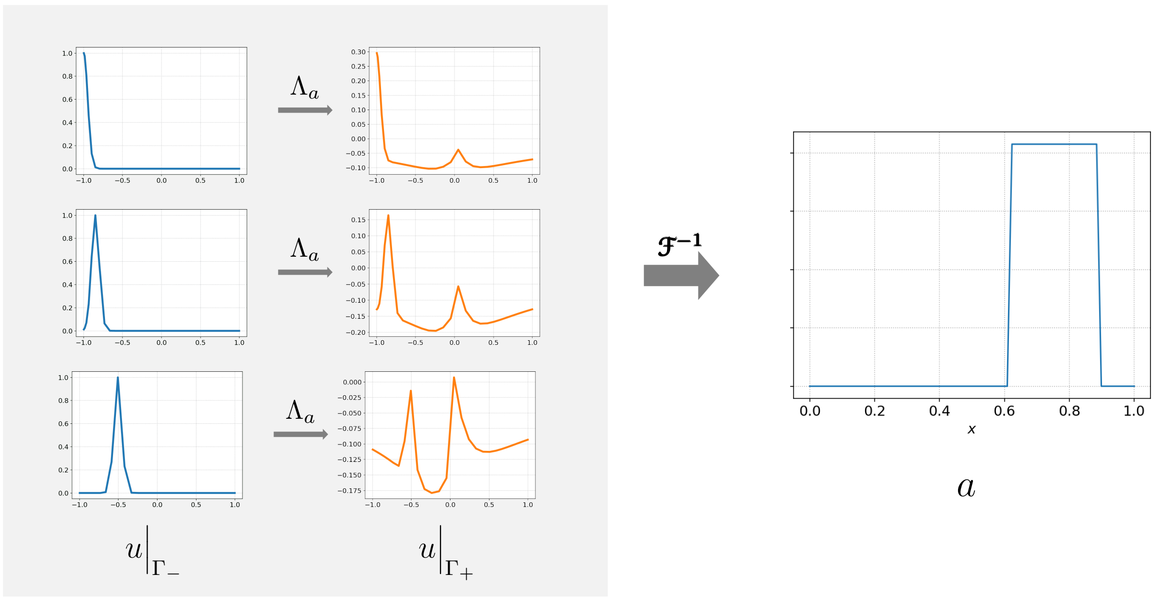


Figure 7.7: Illustration of Optimal Imaging through the Radiative Transport Equation. Left: Input is the Albedo operator (7.25) illustrated with three mappings between the inflow and outflow boundaries. Right: Output is the Scattering coefficient.

with $c = 2\times 10^4/3$. It represents a homogeneous medium with square-shaped inclusions, randomly spread in the domain (see Figure 7.6). Here, $m = \text{Unif}(\{1,2,3,4\})$, and $\{(c_{1,k}, c_{2,k})\} \sim \text{Unif}([0,1]^{m\times 2})$. For each draw of the coefficient, 20 Dirichlet boundary values are prescribed, exactly as in the EIT experiment with trigonometric coefficients. The corresponding (approximate) solutions of the Helmholtz equation (7.22) are computed with a central finite difference scheme, and the Neumann trace is evaluated to represent the DtN map.

7.4.3 Radiative Transport Equation and Optical Imaging.

Next, we consider the radiative transport equation (7.24) in the domain $X \times V$, where $X = [0,1]$ and $V = [-1,1]$, with $\varepsilon = 1$. Consequently,

$$\Gamma_- = \{(0,v) : v \in [0,1]\} \cup \{(1,v) : v \in [-1,0]\}.$$

The task is to infer the absorption and scattering coefficients from the Albedo operator (7.25). To this end, we fix $k(v,v') = 1$, $\sigma_a = 1-a$ in (7.24) and draw the absorption coefficient a from the distribution,

$$a(x) = c\chi_{[-1/2,1/2]}(rx - x_0) + 1,$$

with χ denoting the characteristic function and with $c \sim \text{Unif}([0.5,1])$, $x_0 \sim \text{Unif}([0,1])$ and $r \sim \text{Unif}([0,0.8])$. Once the coefficient is drawn, boundary conditions on the inflow boundary Γ_- are imposed by setting

$$\phi_\ell(0,v) = \exp\big(-200\,(v-v_\ell)^2\big), \quad \phi_\ell(1,v) = 0, \quad v_\ell > 0,$$

and

$$\phi_\ell(0,v) = 0, \quad \phi_\ell(1,v) = \exp\big(-200\,(v-v_\ell)^2\big), \quad v_\ell < 0,$$

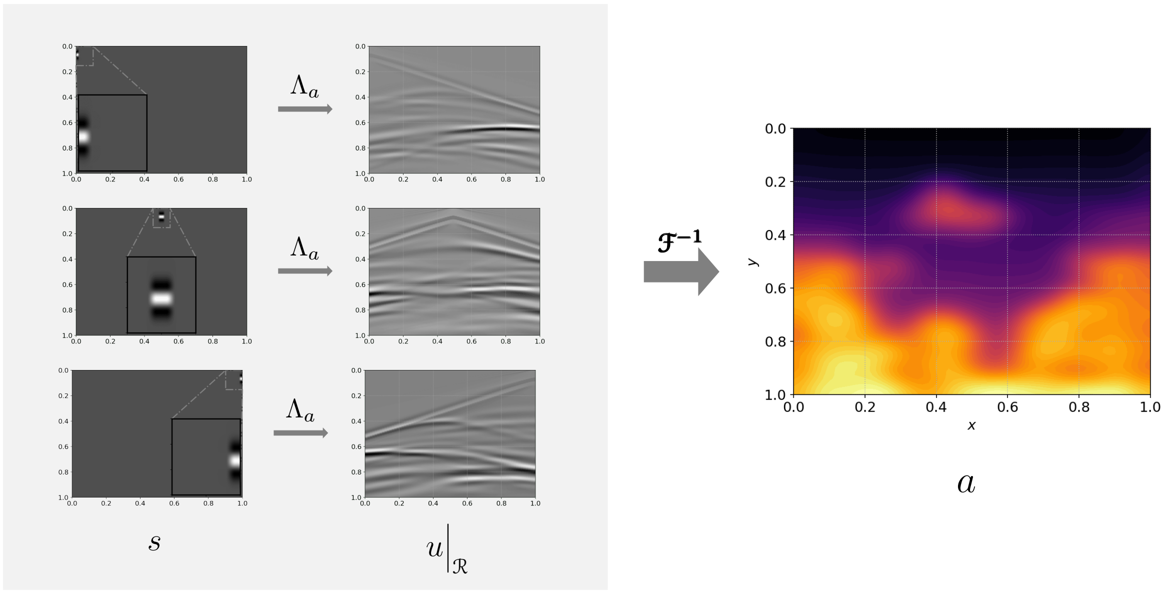


Figure 7.8: Illustration of Seismic Imaging. Left: Input is Source-to-Receiver map (7.28) between Incident waves generated at Sources to Temporal signals recorded at Receivers. Right: Output is the velocity coefficient, corresponding to *Style A* dataset of [110].

with v_ℓ being the ℓ-th quadrature point used to approximate the integral term in (7.24), $1 \leq \ell \leq 32$. Then, the radiative transport equation is approximated with a finite-element method, and the resulting solution u_ℓ is evaluated at the outflow boundary Γ_+ as the output of the Albedo operator (7.25).

7.4.4 Seismic Imaging.

In the final test, we model seismic imaging by considering the acoustic wave equation (7.27) in the space-time domain $[0,1]^2 \times [0,T]$ and the task at hand is to learn the underlying squared-slowness coefficients a from the *source-to-receiver* map (7.28). To this end, we choose two types of coefficients from [110], the so-called *Style-A* and *CurveVel-A* datasets. For each medium, waves are generated at source locations $(x_{s_\ell,0})$ on the vertical boundary, for $\ell = 1,\dots,5$. The corresponding acoustic wave equation is solved with a finite difference scheme, and the temporal data is recorded at receivers on the vertical boundary.

7.4.5 Numerical Results

We start by testing the proposed architecture and baselines on a test set

$$\mathcal{S}_{test} = \left\{ \left(\{\Psi_\ell^{(i)}\}_{\ell=1}^{L}, a^{(i)} \right),\ i = 1, ..., M \right\} \tag{7.41}$$

with the input measure μ_Ψ discretized as the training input measure (observe that not only the number of samples L is the same, but also their ordering). The results, together with the number of testing samples M, are summarized in Table 7.1.

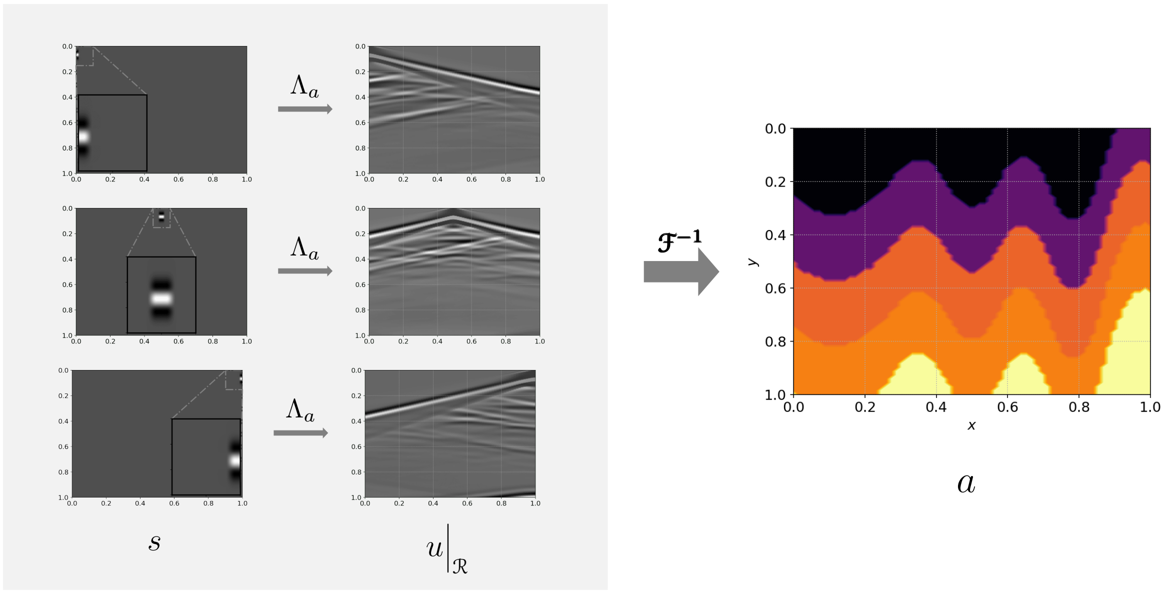


Figure 7.9: Illustration of Seismic Imaging. Left: Input is Source-to-Receiver map (7.28) between Incident waves generated at Sources to Temporal signals recorded at Receivers. Right: Output is the velocity coefficient, corresponding to *CurveVel A* dataset of [110]

As the table shows, NIO is the best-performing model, outperforming the next-best FCNN model by almost halving the errors for most of the benchmarks. We also observe that even for the Seismic Imaging problem, NIO is either outperforming or on par with FCNN. This is particularly noteworthy as the FCNN architecture was demonstrated to be one of the states of the art on this problem in [110] among several machine learning models.

The low test errors are futher visually reinforced. In Figure 7.10, we show two randomly drawn test samples for the Calderón Problem for inferring conductivity with trigonometric coefficients by EIT. For both these test samples, we see that NIO (and FCNN) can accurately approximate the ground truth without any visible artifacts. This observation correlates with very small test errors with NIO. At least for these two samples, there appears to be little visible difference between NIO and FCNN. Nevertheless, the results from Table 7.1 demonstrate that NIO outperforms FCNN considerably on this problem by almost halving the test error.

Next, in Figure 7.11, we focus on the discontinuous Heart&Lungs Phantom inferred with EIT. Also, in this case, there is no visual difference between the NIO and FCNN, which are both very accurate in reconstructing the ground truth, and this is indeed consistent with the very low generalization error achieved by both models.

In Figure 7.12, we plot the results of two randomly chosen test samples for the inverse wave scattering problem and compare the ground truth with the reconstruction with NIO and FCNN. In the first sample (top row), both models accurately reconstruct the ground truth coefficient with very little visible difference between the competing models. In contrast, in the second sample, the reconstruction with NIO and FCNN show noticeable differences. In particular, FCNN cannot reconstruct the small rectangular scatterer (in the middle of the square domain), whereas NIO can reconstruct it. This possibly explains why NIO is significantly more accurate (see Table 7.1) for this experiment in reconstructing scatterers.

			DONet		**FCNN**		**NIO**	
	N	M	$L^1\downarrow$	$L^2\downarrow$	$L^1\downarrow$	$L^2\downarrow$	$L^1\downarrow$	$L^2\downarrow$
Calderón Problem Trigonometric	4096	2048	1.97%	2.36%	1.49%	1.82%	0.85%	1.05%
Calderón Problem Heart&Lungs	4096	2048	0.95%	3.69%	0.27%	1.62%	0.18%	1.16%
Inverse Wave Scattering	4096	2048	3.83%	7.41%	2.53%	7.55%	1.07%	2.94%
Radiative transport	4096	2048	2.35%	4.35%	1.46%	3.71%	1.1%	2.9%
Seismic Imaging CurveVel - A	55000	6000	3.98%	5.86%	2.65%	5.05%	2.71%	4.71%
Seismic Imaging Style - A	22000	5000	3.82%	5.17%	3.12%	4.63%	3.04%	4.36%

Table 7.1: Relative median L^1-error and L^2-error computed over M testing samples for different benchmarks and models trained with N training samples.

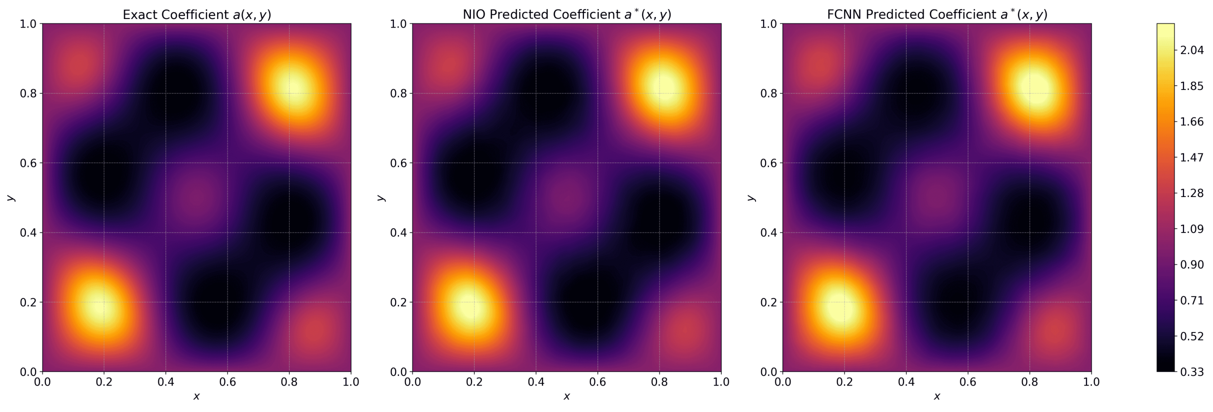
In Figure 7.13, we plot two randomly chosen test samples to recover the absorption coefficient with optical imaging for the Radiative transport equation (7.24). The ground truth and reconstructions obtained with NIO and FCNN are shown. For the first test sample, both models can provide an accurate reconstruction with a sharp resolution of the discontinuities in the absorption coefficient. On the other hand, for the second sample (Figure 7.13 Right), we see that FCNN gets the correct location but the wrong magnitude of the discontinuity, whereas NIO can approximate both accurately, probably accounting for the significant gain in accuracy on this problem (see Table 7.1 of main text).

In Figures 7.14 and 7.15, we show two randomly chosen test samples for Seismic imaging of the subsurface property (squared slowness) by the acoustic wave equation (7.27), corresponding to the *CurveVel-A* and *Style A* datasets (considered in [110]), respectively. Both figures show that NIO and FCNN reconstruct the coefficient reasonably accurately, although slight differences exist between the models.

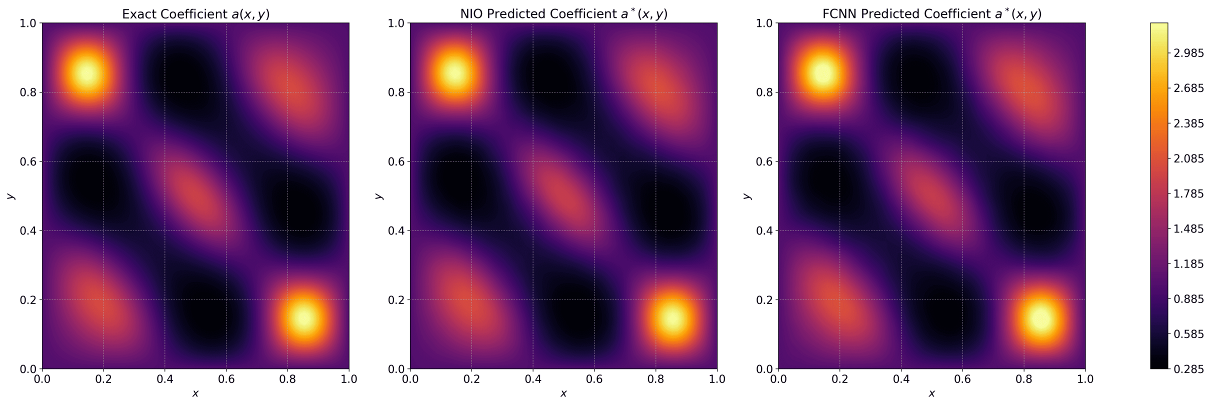
Robustness of Reconstruction to Λ_a-Discretizazion.

As outlined in Section 7.3.4, one crucial property that the model should exhibit is robustness to the number of samples $\tilde{L}$ used to approximate the pushforward measure μ_Ψ. To assess this, we conduct two different experiments.

In the first experiment, we consider the test set 7.41 and for each benchmark, we construct a new test set by picking at random (without replacement) $\tilde{L}$ samples from $\{\Psi_\ell\}_{\ell=1}^{L}$, $\tilde{L} \leq L$, and compute the corresponding testing error (for NIO and baselines). As a remark, it should be noted that NIO can be evaluated directly for any input $\{\Psi_k\}_{k=1}^{\tilde{L}}$, without any change in the architecture. On the other hand, in order to even evaluate the baselines, interpolation (we chose to use the nearest interpolation) must be used to obtain inputs consisting of exactly L samples. In Figure 7.16, we plot the median L^1-error obtained as a function of the number of samples $\tilde{L}$ for different models and benchmarks. We observe that the performance of NIO remains invariant with respect to $\tilde{L}$, with the testing error only showing a *slight* increase as $\tilde{L}$ decreases, which is typically expected since the approximation of the measure μ_Ψ by the empirical distribution becomes less accurate as the number of samples decreases. In contrast, the



(a) Test Sample 1



(b) Test Sample 2

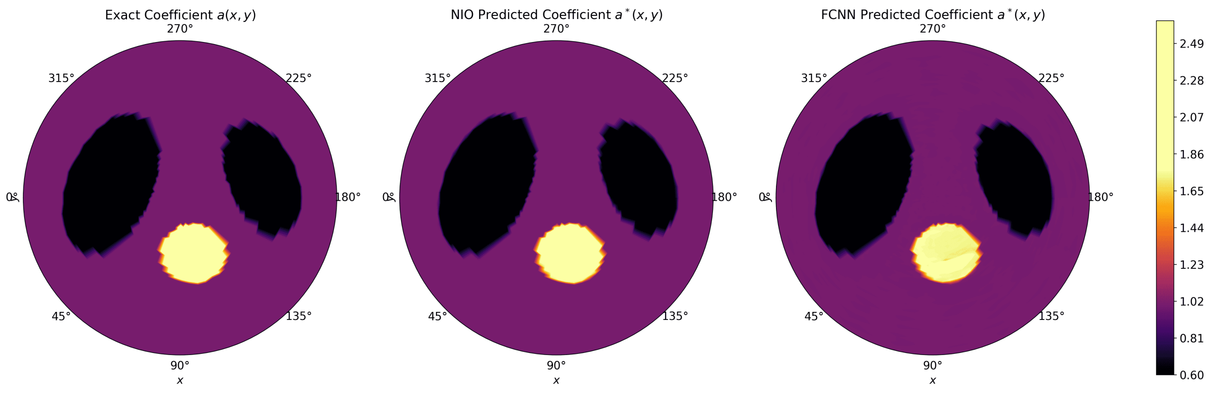
Figure 7.10: Exact and predicted coefficients for two different test samples (Rows) for the Calderón problem with Trigonometric coefficients. Left Column: Ground Truth. Middle Column: NIO reconstruction. Right Column: FCNN Reconstruction.

performance of the baselines significantly deteriorate when the number of samples $\tilde{L}$ is different from the training ones L. The same behaviour is observed even when $\tilde{L}$ is extremely close to L.

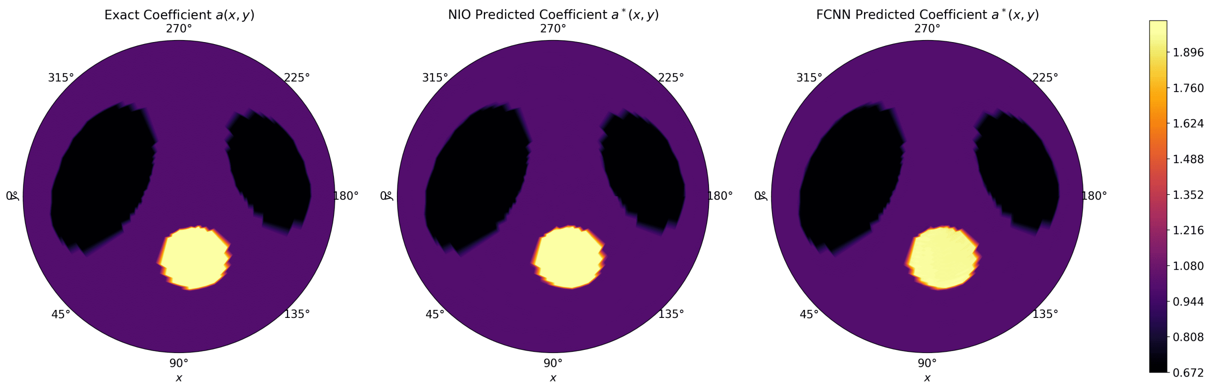
Next, we consider the Calderón problem with trigonometric functions and the inverse wave scattering problem. We generate a new testing set from scratch, consisting of input-output pairs $(\{\Psi_\ell\}_{\ell=1}^{L}, a)$, where $L = 100$ (compared to the $L = 20$ samples used for model training). We then conduct the same experiments as before and present the results in Figure 7.17. These results further reinforce the fact that NIO exhibits invariance with respect to the discretization of the input measure μ_Ψ.

Robustness of Reconstruction to Noise

Inverse problems, such as the abstract PDE inverse problem (2.16), can be very sensitive to noise as the stability estimate (2.17) indicates, and reconstruction methods have to show some robustness with respect to noisy measurements in order to be practically useful. To test the robustness of NIO (and competing models) to noise, we take all the benchmark test problems reported in Table 4.4 of the main



(a) Test Sample 1

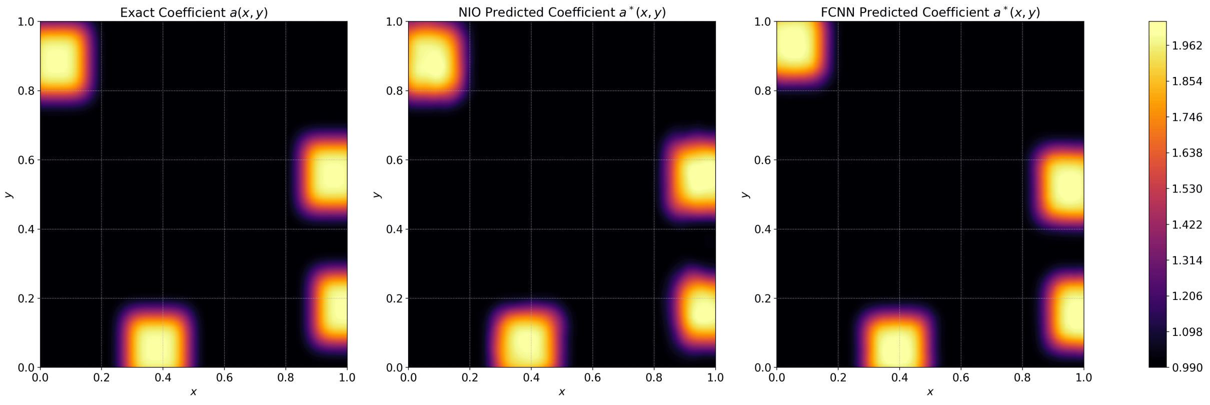


(b) Test Sample 2

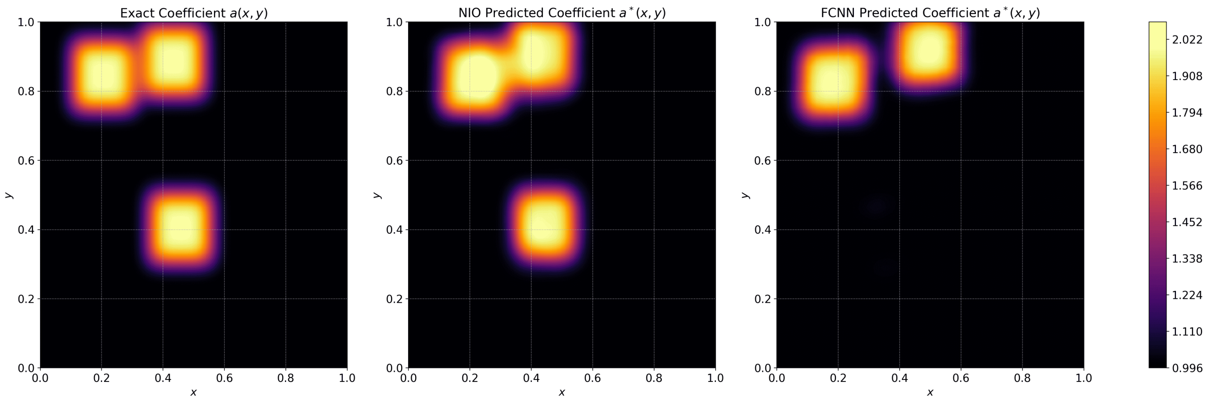
Figure 7.11: Exact and predicted coefficients for two different test samples (Rows) for the Calderón problem with Heart&Lungs Phantom. Left Column: Ground Truth. Middle Column: NIO reconstruction. Right Column: FCNN Reconstruction.

text and add 1% noise to the inputs to each model at test time. Table 7.2 presents the resulting test errors. This table shows that NIO (as well as DOnet and FCNN) is very robust to this measurement noise.

Furthermore, upon closer examination of the results, it is evident that the models displaying the highest robustness with respect to additional noise are those trained using the *log-MinMax* data scaling transformation. To validate this observation, we consider the inverse wave scattering problem and train NIO with the hyperparameters reported in Table 9.4, but employing the *log-MinMax* scaling of the data. Additionally, instead of monitoring the validation error computed on the noiseless data, we monitor the validation error computed on data corrupted by 10% noise and interrupt the training based on this metric. The final median testing error on 1%-noisy data is 1.64%, two times lower than the value reported in Table 7.2. Moreover, the testing error on the noise-free data only marginally increased to 1.61%. These findings suggest that utilizing *log-MinMax* scaling and potentially monitoring the validation error on the corrupted data can significantly enhance the model's robustness to noise, with minimal loss in performance on the noise-free data.



(a) Test Sample 1

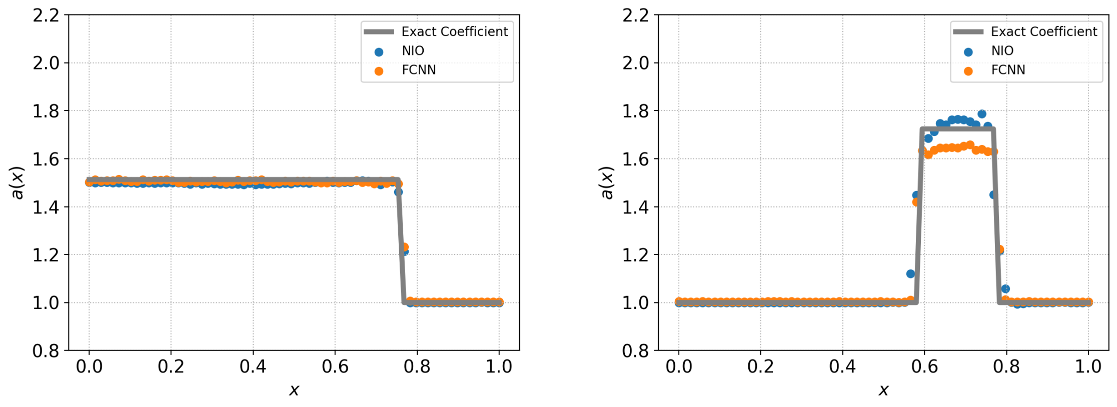


(b) Test Sample 2

Figure 7.12: Exact and predicted coefficients for two different test samples (Rows) for the Inverse Wave Scattering with Helmholtz Equation. Left Column: Ground Truth. Middle Column: NIO reconstruction. Right Column: FCNN Reconstruction.

Robustness of Reconstructions to Varying Grid Sizes.

Although the inputs and outputs to the inverse problem (2.16) are continuous objects in principle, in practice, one has to deal with discretized versions of both inputs and outputs. This is true when the ground truth is generated by numerical simulations and observed through other forms of measurement. It is highly desirable that an operator learning algorithm be robust to the resolutions at which it is tested; see [32] for further discussion on this topic. To test if the proposed NIO architecture is robust with respect to resolution, we focus on the inverse wave scattering with the Helmholtz equation example, where NIO was trained with data obtained from a finite difference scheme on a uniform 70×70 grid. To test the robustness with respect to resolution, we use this trained model to also infer at two different resolutions, namely at 50×50 and 100×100, and present the results, together with DeepONet and FCNN baselines in Table 7.3 to observe that NIO (and the baselines) is robust to varying resolutions.



(a) Test Sample 1 (b) Test Sample 2

Figure 7.13: Exact and predicted absorption coefficients for two different test samples, obtained with optical imaging for the radiative transport Equation.

Robustness of Reconstruction to Random Sensors Location.

While training data typically assumes equidistant placement of sensors along the boundaries of the square domain (as they are synthetically generated using standard numerical methods), real-world scenarios often involve sensors located randomly along the boundaries. Hence, the learning model must exhibit robustness to these random sensor placements.

Our experiments to assess this robustness focus on two specific problems: the Calderón problem with trigonometric function and inverse wave scattering. We perform testing with input data obtained from 200 sensors randomly distributed along the domain boundary. Observe that the training data accounts for 272 sensors. Therefore, before feeding the data to NIO (and baselines), we interpolate it onto the original equispaced set of points. For both problems, we examine the L^1 error, as presented in Table 7.4.

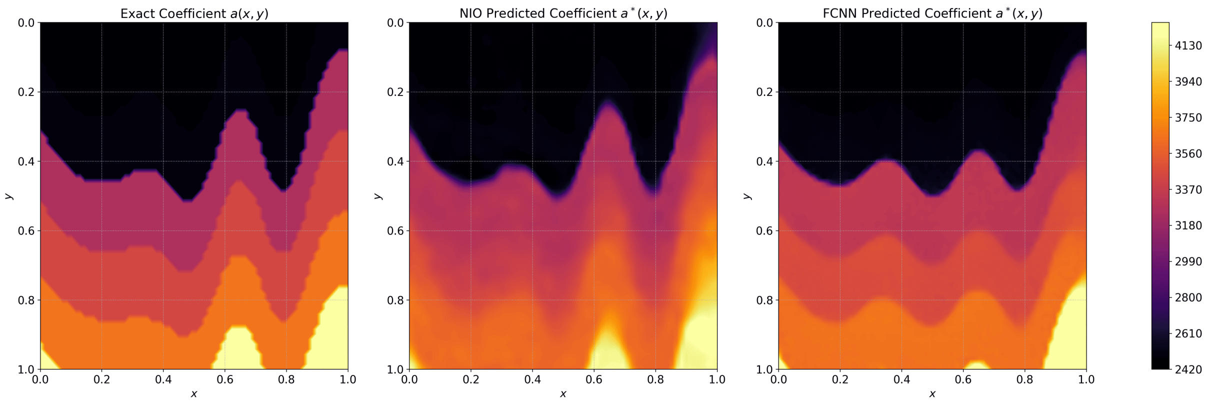
The results show that the L^1-error increases only to 1.18% and 1.43%, compared to the original setup where the errors were 0.86% and 1.11%, respectively. These findings underscore the NIO model's ability to maintain robust performance even when the boundary sensors are placed at different locations.

Out-of-Distribution Reconstruction.

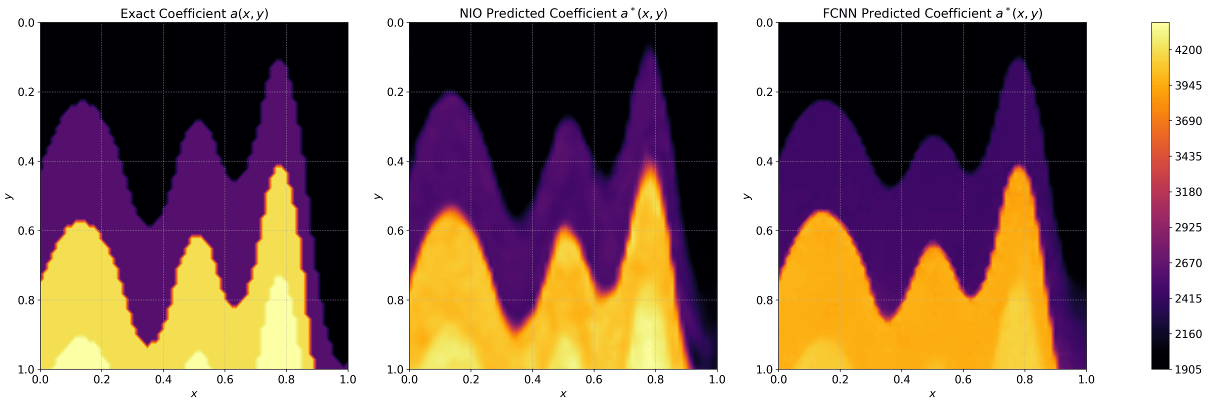
In addition to *in-distribution* testing, we also consider an *out-of-distribution* testing task. This will enable us to evaluate the ability of the models to *generalize* to inputs (and outputs) that possess different features from the training ones.

First, we considered the Calderón Problem (Trigonometric) benchmark. The coefficients in the training distribution were sampled from the exponential of a sum of sines, with up to 4 frequency modes (up to 8π). We now test with the following distributions:

- Distribution A: $a(x,y) = \exp\left(\sum_{k=1}^{6} c_k \sin(k\pi x)\sin(k\pi y)/k^{\frac{3}{2}}\right)$
- Distribution B: $a(x,y) = \exp\left(\sum_{k=1}^{6} c_k \sin(k\pi x)\sin(k\pi y)/k\right)$

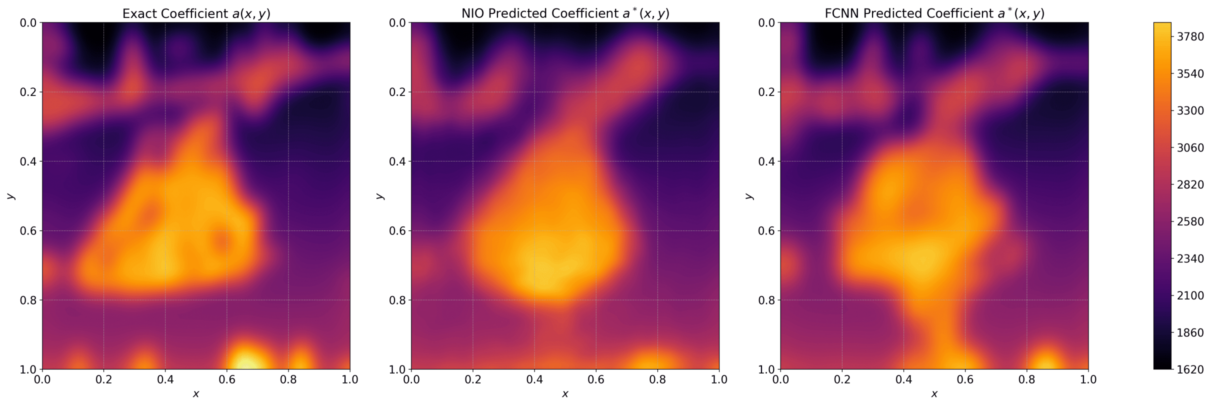


(a) Test Sample 1

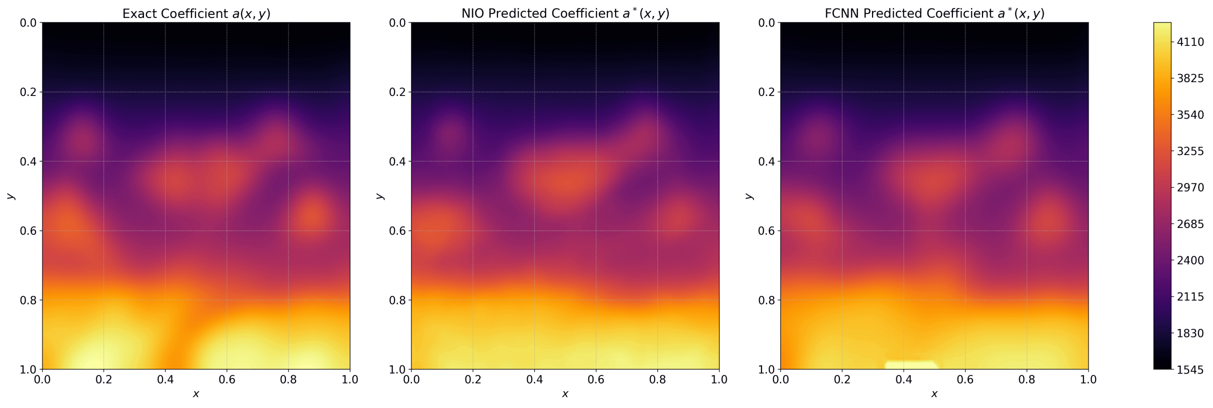


(b) Test Sample 2

Figure 7.14: Exact and predicted coefficients for two different test samples (Rows) for the Seismic Imaging with the acoustic wave equation with *CurveVel A* data set. Left Column: Ground Truth. Middle Column: NIO reconstruction. Right Column: FCNN Reconstruction.

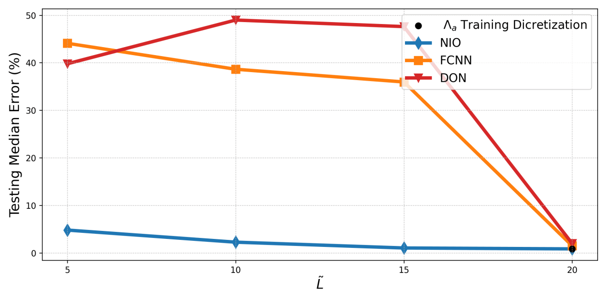


(a) Test Sample 1

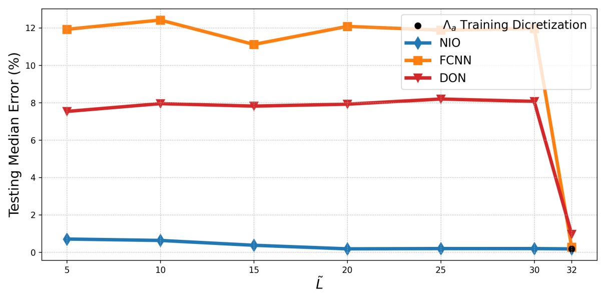


(b) Test Sample 2

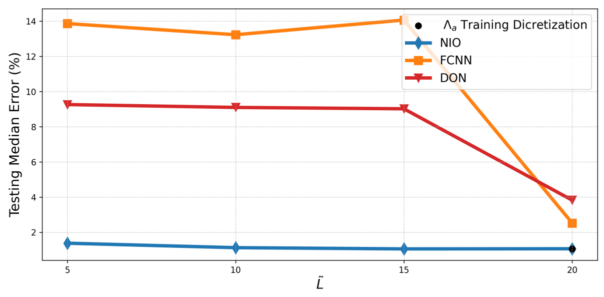
Figure 7.15: Exact and predicted coefficients for two different test samples (Rows) for the Seismic Imaging with the acoustic wave equation with *Style A* data set. Left Column: Ground Truth. Middle Column: NIO reconstruction. Right Column: FCNN Reconstruction.



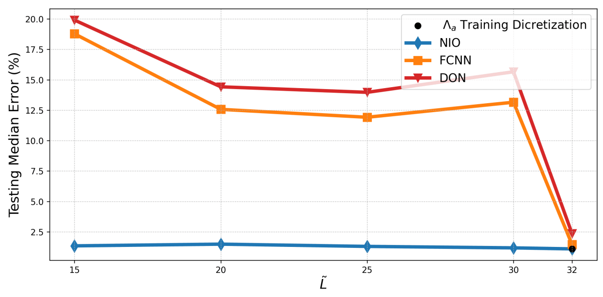
(a) Calderón Problem Trigonometric



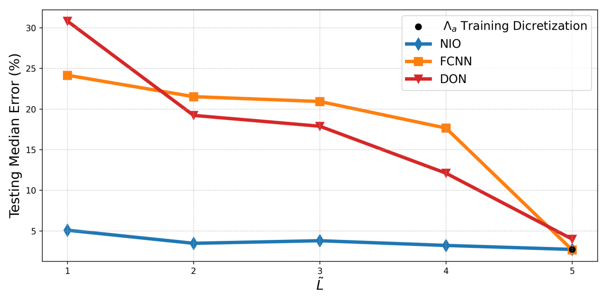
(b) Calderón Problem Heart&Lungs



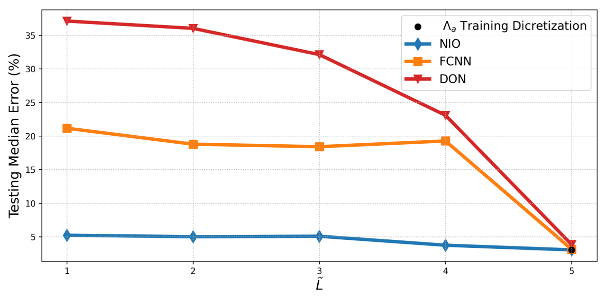
(c) Inverse Wave Scattering



(d) Radiative transport

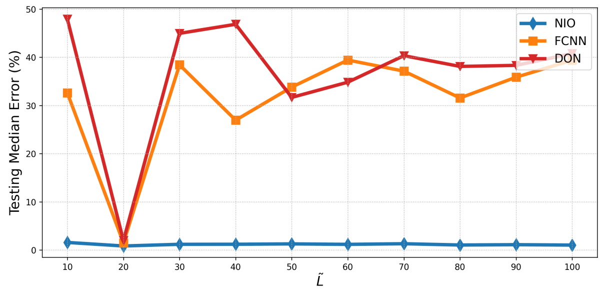


(e) Seismic Imaging CurveVel - A

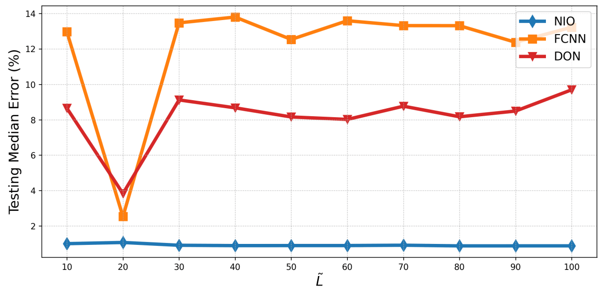


(f) Seismic Imaging StyleVel - A

Figure 7.16: Median of the L^1-error computed over testing samples $(\{\Psi_k\}_{k=1}^{\tilde{L}}, a)$ VS $\tilde{L}$ for different benchmarks with different models ($\tilde{L} < L$).



(a) Calderón Problem Trigonometric



(b) Inverse Wave Scattering

Figure 7.17: Median of the L^1-error computed over testing samples $(\{\Psi_k\}_{k=1}^{\tilde{L}}, a)$ VS $\tilde{L}$ for different benchmarks with different models ($\tilde{L}$ spans the entire range 10-100).

	DONet	**FCNN**	**NIO**
Calderón Problem Trigonometric	2.02%	1.51%	0.91%
Calderón Problem Heart&Lungs	0.95%	0.27%	0.18%
Inverse Wave Scattering	3.83%	2.54%	3.72%
Radiative transport	2.38%	1.47%	1.1%
Seismic Imaging CurveVel - A	3.98%	2.65%	2.73%
Seismic Imaging Style - A	3.82%	3.13%	3.09%

Table 7.2: Median of the relative L^1-error computed over 1%-noisy testing samples for different benchmarks with different models.

Here, c_k is a uniformly distributed random variable in the range $[0,1]^m$. The coefficients include up to 6 frequency modes, with different decays of the higher order modes (1.5 and 1). Thus, at test time, the model now has to infer data with significantly higher frequencies than the training data.

Next, we consider the Calderon problem of the Heart&Lungs. Here, the training distribution was based on a normally distributed perturbation of the Heart&Lungs Phantom, with the amplitude of the perturbation being at most 8% of the Phantom values (See 7.5). At test time, we now consider perturbations with amplitudes being 12% of the Phantom values, thus sampling from a different distribution. For instance, this higher amplitude could model individuals with some diseases. Note that even higher variations are probably unrealistic as this problem models body organs and one has to restrict to some biological constraints.

As a third benchmark, we consider the inverse scattering problem. Here, the training distribution was of a coefficient that consisted of between 1-4 scatterers, *of identical shape*, whose locations were randomly chosen (See 7.6). We chose to test this model now on two different test distributions

- Distribution A: $a(x,y) = \sum_{k=1}^{5} \exp\big(-c(x-c_{1,k})^4 - c(y-c_{2,k})^4\big)$. In this case, we use a family of

	DONet		**FCNN**		**NIO**	
Resolution	50×50	100×100	50×50	100×100	50×50	100×100
Inverse Wave Scattering	3.74%	3.63%	1.81%	1.66%	0.93%	0.95%

Table 7.3: Relative median L^1-error computed over testing samples generated at different resolutions (grid sizes).

	DONet	**FCNN**	**NIO**
	$L^1 \downarrow$	$L^1 \downarrow$	$L^1 \downarrow$
Calderón Problem Trigonometric	3.39%	1.88%	1.18%
Inverse Wave Scattering	3.84%	2.53%	1.43%

Table 7.4: Relative median L^1-error computed over testing samples with random location of the boundary measurements with different models and different benchmarks.

coefficients with a fixed number of inclusions equal to five.

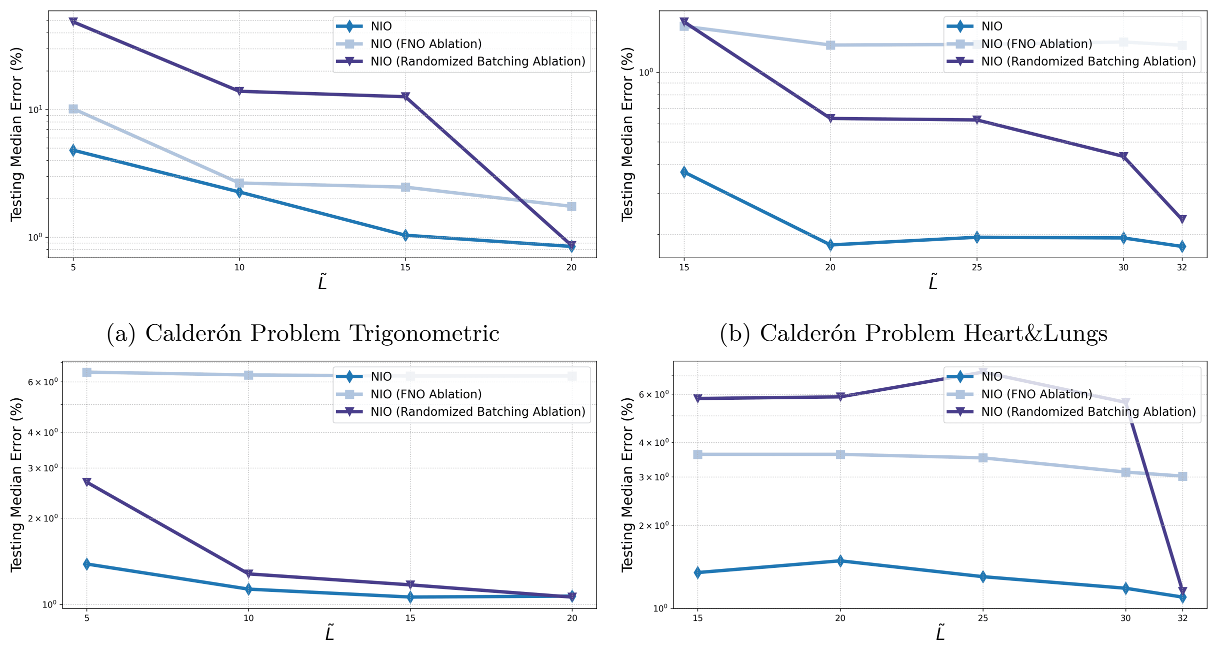
- Distribution B: $a(x,y) = \sum_{k=1}^{m} \exp\big(-b_k^4(x-c_{1,k})^4 - b_k^4(y-c_{2,k})^4\big)$, with $b_k \sim \mathrm{Unif}[5,15]$, $m \sim \mathrm{Unif}(\{1,2,3,4\}$ and $\{(c_{1,k},c_{2,k})\} \sim \mathrm{Unif}([0,1]^{m\times 2})$. This corresponds to a medium with one to four scatterers with varying shapes.

The relative median L^1 error for different models and different out-of-distribution testing is reported in Table 7.5. We observe that NIO generalizes well to unseen data, with test errors increasing at most by approximately a factor of 4, and still outperforms the baselines in all cases.

	DONet	**FCNN**	**NIO**
	$L^1 \downarrow$	$L^1 \downarrow$	$L^1 \downarrow$
Calderón Problem Trigonometric Distribution A	1.37%	1.27%	1.2%
Calderón Problem Trigonometric Distribution B	1.62%	1.28%	0.91%
Calderón Problem Heart&Lungs	1.05%	0.28%	0.19%
Inverse Wave Scattering Distribution A	4.61%	3.84%	3.0%
Inverse Wave Scattering Distribution B	8.61%	8.98%	4.54%

Table 7.5: Relative median L^1-error computed over out-of-distribution test samples with different models and different benchmarks.

Ablation Studies.



(a) Calderón Problem Trigonometric (b) Calderón Problem Heart&Lungs

(c) Inverse Wave Scattering (d) Radiative Transport

Figure 7.18: Median of the L^1-error computed over testing samples $(\{\Psi_k\}_{k=1}^{\tilde{L}}, a)$, VS $\tilde{L}$ for different benchmarks with different models (NIO and ablations).

We conduct two ablation studies focusing on two key elements of the Neural Inverse Operator. Firstly, we focus on the architecture shown in Figure 7.3, where we remove the nonlinear part $\mathcal{M}$ of FNO and set $d_v = 1$. In this case, we have:

$$a^*(z) = R(f_1, \dots, f_L, z) = \frac{D}{L}\sum_{\ell=1}^{L} f_\ell + Ez, \tag{7.42}$$

where E and D are real-valued parameters. With these experiments, we aim to assess if the channels' mixing realized with $\mathcal{M}$ could improve the model's performance. Secondly, we examine the influence of randomized batching by training NIO without including the algorithm.

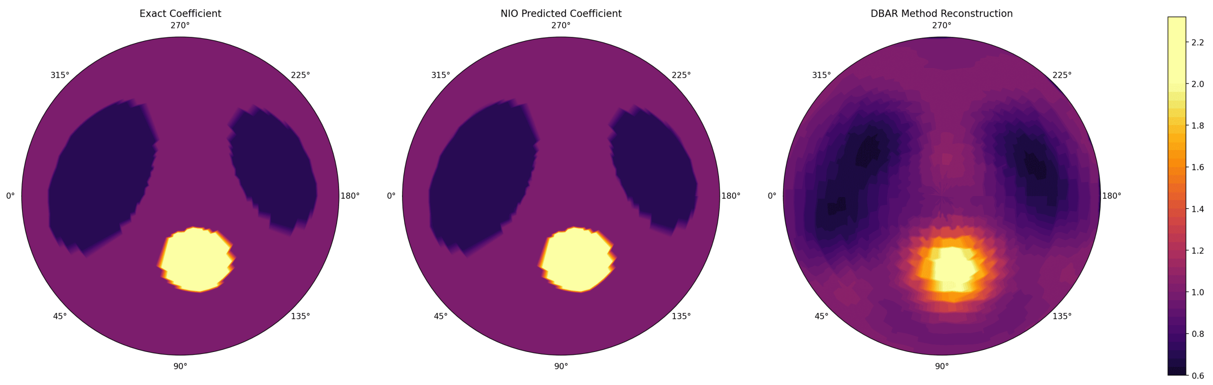
In order to maintain consistency, we use the same hyperparameter configurations for the ablation models as those of the best-performing NIO models (refer to Table 9.4 for the specific values).

The ablation study is being carried out for the Calderón problem with trigonometric function and heart&lungs Phantom, inverse wave scattering, and radiative transport. We consider the same experimental setup outlined in Section 7.4.5 and report the corresponding results in Figure 7.18. The figure shows that the nonlinear term $\mathcal{M}$ and randomized batching significantly improve the model's performance. For the Calderón problem with trigonometric coefficients, removing $\mathcal{M}$ resulted in a nearly 2-times increase of the generalization error. On the other hand, for the remaining problems, the improvement is considerably more relevant, up to a factor of six for the inverse wave scattering.

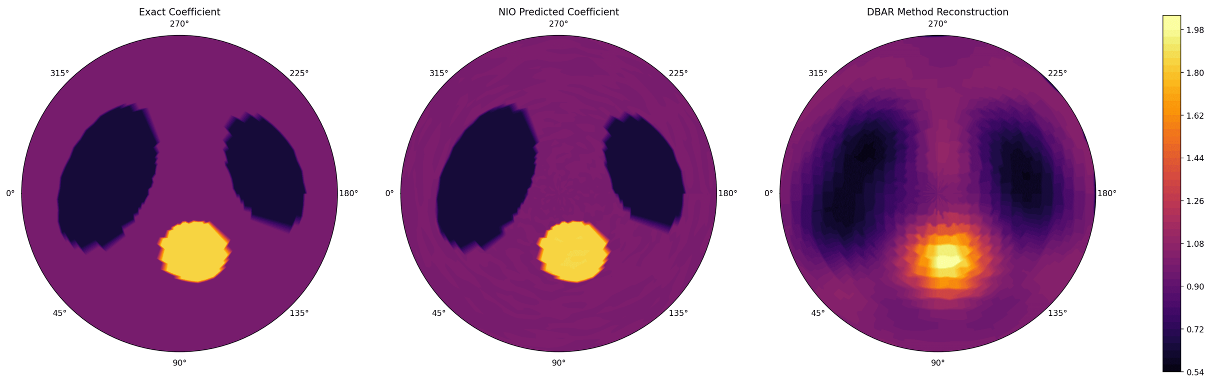
It should be noted that in all experiments, only between $L = 20$ and $L = 32$ boundary measurements are used for training. In this scenario, the limited measurement data represents a bottleneck in accurately reconstructing the target coefficient. With a larger number, the nonlinear part may lead to an even greater reduction in the generalization error.

Regarding randomized batching, the improvements for the Calderón problem and radiative transport are substantial, ranging from 5 to 10 times compared to the ablated version. On the other hand, the improvements are more modest for the Inverse Wave Scattering, and the algorithm does not appear to considerably enhance generalization. However, these results are particularly impressive on account of the virtually zero cost associated with the randomized batching algorithm.

7.4.6 Comparison with Standard Numerical Methods for Inverse Problems



(a) Test Sample 1



(b) Test Sample 2

Figure 7.19: Exact and predicted coefficients for two different test samples (Rows) for the Calderón problem with Trigonometric-coefficients. Left Column: Ground Truth. Middle Column: NIO reconstruction. Right Column: Reconstruction with the D-bar Direct method of [115].

In this section, we compare the performance of the proposed architecture, in terms of accuracy and inference time, with standard numerical methods, particularly PDE-constrained optimization techniques.

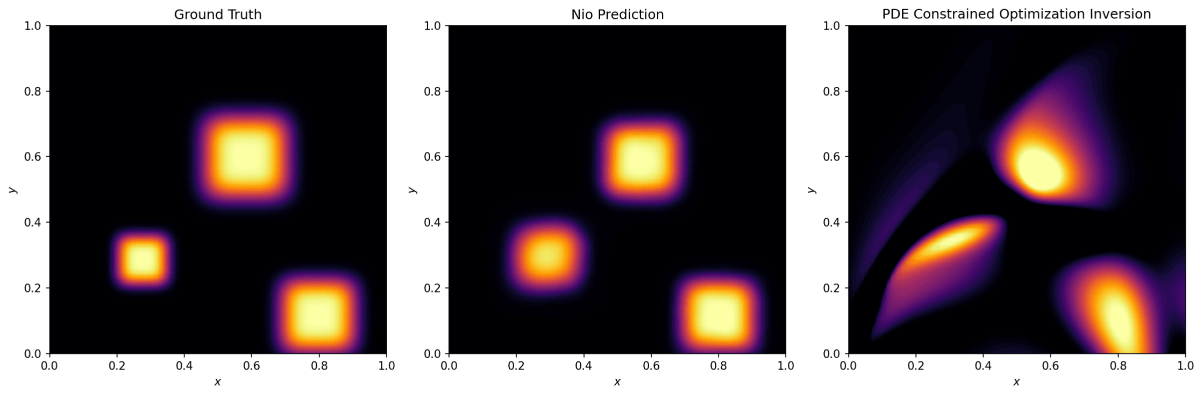


Figure 7.20: Exact and predicted coefficients for an out-of-distribution test sample for the Inverse Wave Scattering. Left Column: Ground Truth. Middle Column: NIO reconstruction. Right Column: Reconstruction with the PDE-constrained optimization method.

Calderón Problem Heart&Lungs

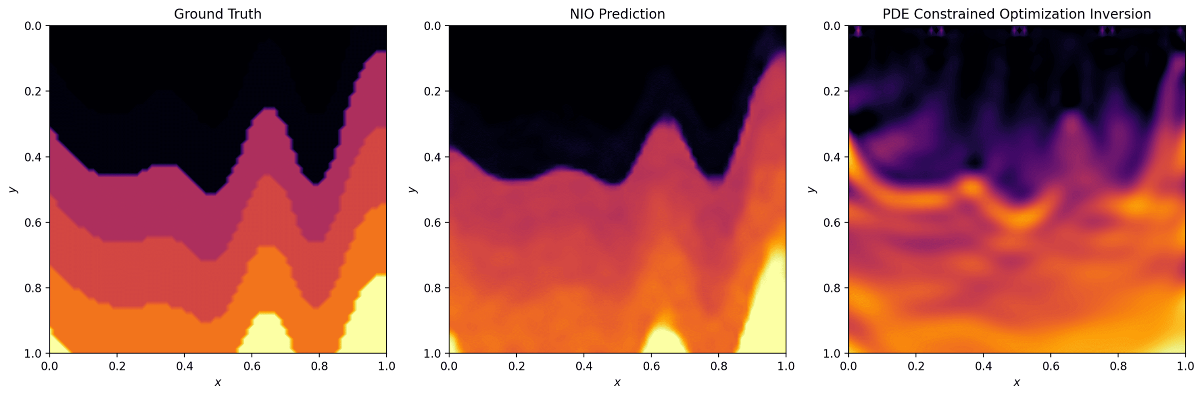
We begin by considering the Calderón problem for the discontinuous Heart&Lungs Phantom and conduct a comparative analysis between the proposed approach and the well-known D-bar method [115], which is commonly employed in the context of EIT. In Figure 7.5, we present the ground truth along with the NIO reconstruction for a randomly selected set of test samples. For comparison, we also include the reconstruction obtained using the D-bar method. As the figure shows, NIO reconstructs the ground truth to very high accuracy, consistent with the very small errors presented in Table 4.4. On the other hand, the D-bar method is relatively inaccurate and provides a blurred and diffusive reconstruction of the shapes. In fact, the L^1-test error for the D-bar method is an unacceptably high 8.75%, compared to the almost 0.15% test error with NIO. This is even more impressive when one looks at the run times. The D-bar method takes approximately 2 hours to run for a single sample, whereas the inference time for NIO is only 0.1 seconds (on CPU). Thus, we can provide a method which *two orders of magnitude more accurate while being four orders of magnitude faster to run.* This highlights the massive gain in performance with machine learning-based methods, such as NIO, compared to traditional direct methods.

Inverse Wave Scattering

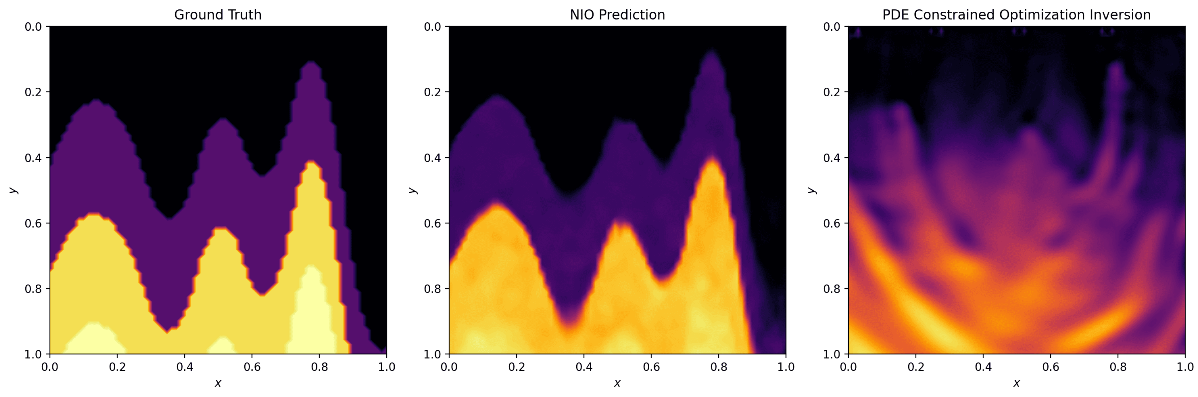
Next, we investigate the Inverse Wave Scattering problem and compare the NIO reconstruction with the results obtained through PDE-constrained optimization. We examine an out-of-distribution (Distribution A) test sample to accomplish this.

For PDE-constrained optimization, we employ a feed-forward neural network with trainable parameters θ that parameterizes the coefficient a. The neural network architecture follows the form specified in Equation 2.24, with L layers, d neurons, and activation function σ. The model parameters θ are initialized randomly, and the Helmholtz equation (7.22) is numerically solved for $L = 20$ realizations of the boundary conditions g_ℓ, where $\ell = 1, ..., L$ and the L^2-loss

$$J(\theta) = \sum_{\ell=1}^{L} ||\Psi_\ell - \tilde{\Psi}_\ell||_2^2 \tag{7.43}$$



(a) Test Sample 1



(b) Test Sample 2

Figure 7.21: Exact and predicted coefficients for two different test samples (Rows) for the Curve Vel family. Left Column: Ground Truth. Middle Column: NIO reconstruction. Right Column: Reconstruction with the PDE-constrained optimization method.

computed. Here, $\tilde{\Psi}_\ell$, $\ell = 1, ..., L$ denotes the target data. The trainable network parameters are updated using LBFG in the direction of the negative gradient of $J(\theta)$. The process is iterated until convergence (1000 iterations, corresponding to 1000×20 PDE solves).

Observe that the hyperparameters of the network are chosen through a random search (as described in the appendix 9.2.2) by picking the one minimizing $J(\theta)$.

The reconstructed coefficient obtained through PDE-constrained optimization, along with the NIO prediction obtained by directly feeding the model with $\tilde{\Psi}_\ell$ for $\ell = 1, \dots, L$, are depicted in Figure 7.20. The NIO reconstruction is significantly more accurate than the one obtained with PDE-constrained optimization, further reinforced by the corresponding L^1-error, equal to 2.3% and 11.1%, respectively. The corresponding total time required to reconstruct the coefficient, amounts to less than 1 second (on CPU) for NIO and 8.5 hours for the traditional method. It is worth noting that the finite difference (FD) solver employed for solving the equation is implemented on GPU within the PyTorch framework. By solving the PDE in parallel for $L = 20$ boundary measurements on 20 different GPUs, the computation time could potentially be reduced to 30 minutes, which is still three orders of magnitude slower than NIO's inference.

Finally, an alternative approach to directly learning the inverse map would be to replace the forward solver with a learning model in the iterative loop described above. Although the surrogate model might achieve low errors in learning the forward map (compared to the inverse map), the inference time for the model would still be approximately 0.1 seconds (on a CPU). If we require 1000 calls to the model for the optimization process, this will result in an overall inference time that is two orders of magnitude longer than NIO. Moreover, the reconstruction accuracy cannot surpass that obtained using the FD solver as forward map, since any surrogate model approximating the forward map will be less accurate than the FD solver itself.

Seismic Imaging

We also tested the PDE-constrained optimization method in the context of Seismic imaging with the CurveVel-A dataset. Unlike the NIO approach, in the PDE-constrained optimization approach, we first discretize the velocity coefficient $a(z)$ using a piecewise-constant parameterization, denoting the parameters as θ. We then solve the wave equation (7.27) using the finite difference method. The method is thus naturally dependent on the discretization of $a(z)$ and the PDE solver used to evaluate the source-to-receiver operator. We use the same experimental setup as provided in [110]. A similar squared L^2 loss to (7.43) is used as the objective function to measure the mismatch between the observed data and the simulated receiver data based on the current prediction of $a(z)$. We also solve a corresponding adjoint wave equation whose solution is used to compute the gradient of the objective function $J(\theta)$ with respect to θ. We use the standard gradient descent method to perform the optimization. Each gradient evaluation requires two time-dependent PDE solves, which is the most expensive part of this PDE-constrained optimization approach.

The method leads to errors (in L^1) 2.5-4 times larger than CNIO for the same test sample while taking approximately 30 minutes of run-time on an 8-core M1-chip CPU. Figure 7.21 depicts the results for two test samples. The L^1-error achieved with NIO is 2.03% and 4.84% for samples 1 and 2, respectively, whereas the ones obtained with PDE-constrained optimization are 5.05% and 16.9%.

8 Conclusions

Physics informed neural networks (PINNs), originally proposed in [59], have recently been extensively used to numerically approximate solutions of both forward and inverse problems for PDEs in different contexts [17, 116, 24] and references therein. One of the main aims of the thesis was to suggest possible explanations for this efficient approximation by PINNs. In particular, one needs to explain why jointly minimizing the PDE residual and the data residual, (stemming from boundary conditions, initial conditions and, possibly noisy, observables of the solution in the case of inverse problems) will lead to bounds (control) on the overall approximation or generalization error.

In Chapter 3 of the thesis, we focused on the forward problem of a wide range of linear and non linear PDEs with sufficiently smooth solutions. We introduced an abstract framework, where an abstract nonlinear PDE is approximated by a PINN. The key point of this algorithm is to minimize the PDE and boundary residuals at training points chosen as the quadrature points, corresponding to an underlying quadrature rule. The resulting generalization error is bounded in the abstract error estimate (3.13). The bound depends on the training error, the number of training samples, and involves constants that stem from stability estimates for the underlying PDE as well as in the quadrature error. From the bound (3.13), one can conclude that as long as the training error is small, the number of quadrature points is sufficiently large, and one has some control on the stability and quadrature constants, the resulting generalization error will be small. A key role in our error analysis is played by the stability of solutions of the underlying PDE that we leveraged to derive the abstract error bound (3.13).

We illustrate our approach with four representative examples for PDEs, i.e, semilinear parabolic PDEs, viscous scalar conservation laws, the incompressible Euler equations of fluid dynamics and dispersive equations. For each of these examples, the abstract framework was worked out and resulted in the bounds (3.28), (3.39), (3.59), (3.76) on the PINN generalization errors. All the bounds were of the form (3.13), in terms of the training and quadrature errors, and with constants relying on stability (and regularity) estimates for *classical* solutions of the underlying PDE. We also presented numerical experiments to validate the proposed theory.

For the viscous scalar conservation laws, we observed very accurate approximations with PINNs, for all values of the viscosity, as long as the underlying solution was at least Lipschitz continuous. However, as expected from the estimate (3.39), the accuracy deteriorated for the inviscid problem, when shocks were formed. This motivated the design of a novel variant of PINNs for the accurate approximation of entropy solutions of scalar conservation laws (Chapter 4). To this end, we based the PDE residual on the weak form of the Kruzkhov entropy conditions (4.7) and solve the resulting min-max optimization problem for determining parameters (weights and biases), for the neural networks approximating the entropy solution as well as test functions. The resulting PINN was termed as a weak PINN (*wPINN*). We presented numerical experiments with the Burgers' equation to illustrate that *wPINNs*, with suitable choices of loss functions and training protocol (see section 4.2), can approximate the entropy solutions of scalar conservation laws accurately. The algorithm is also supported by rigorous theoretical results which are outlined in more details in [71].

In Chapter 5, we focused on inverse problems for PDE, specifically on the so-called *data assimilation* or *unique continuation.* For these problems, the inputs necessary for the forward problem of the underlying PDE (for instance the abstract pde (2.3)), such as initial and boundary conditions are unknown. However, one has access to observables of the solution on a subset of the underlying domain, which is called *observation domain.* Such data assimilation problems arise in many applications, particularly in geophysics and meteorology.

We followed the outline of Chapter 3 and began by posing the data assimilation inverse problem in an abstract framework for the generic PDE (5.9). Under the assumption that solutions to the underlying inverse problem (5.9), (5.10), satisfy a *conditional stability estimate*, we were able to prove a rigorous upper bound on the error (5.18) of the PINN, in terms of the (computable) training error (5.19) and the number of training samples (with a rate prescribed by the quadrature errors). Remarkably, this bound shares the same structure as the one obtained for the forward problem. This observation suggests that, irrespective of whether applied to forward or inverse problems, the mechanism by which PINNs can effectively approximate solutions of such problems remains the same. More specifically, this mechanism leverages the (conditional) stability estimate on the corresponding inverse or forward problem for the underlying PDE.

We then proceeded and illustrated this abstract framework for concrete examples of linear PDEs that arise in a wide variety of models (Poisson, heat, wave and Stokes equations). All these PDEs possess conditional stability estimates for the underlying data assimilation inverse problem, proved either by the well-known *three balls inequalities* or *Carleman estimates.* We adapt the abstract formalism to each example and provide concrete estimates on the generalization error. Numerical experiments are presented and validated the proposed theory for each of the linear PDEs considered here. We observe from the numerical experiments that PINNs are very efficient at approximating the underlying inverse problem. The resulting errors are very small and are less than 1%, even for a few training samples.

A key advantage of machine learning approaches, as highlighted in the introduction, is their ability to serve as fast surrogates, particularly for parametric PDEs. Once the model has been trained on a (random) sampling of the parameter space, it can efficiently infer solutions for other parameters with minimal computational cost. The choice of the loss function to train the model can stem from various sources: (1) solely data, generated by solving the underlying PDEs for different parameter realizations, (2) solely physics-based constraints, or (3) a combination of data and physics. An example of the second approach was presented in Section 3.7.5, where the parameterized KdV equation was solved with a physics-informed loss function in an unsupervised manner. In the same spirit, in Section 6.1 we considered the parametrized version of the heat equation 7.1 and solved it for different dimensions of the underlying parameter space (up to 100), showing a growth of the generalization error in terms of the parameter space dimension between linear and quadratic.

However, in cases where the high dimensionality of the PDE arises from the large state space, the data-driven approach is often not feasible due to the *curse of dimensionality* affecting traditional numerical methods. On the other hand, physics-informed neural networks (PINNs) offer a valuable approach to solving PDEs with large state spaces. This has been theoretically justified in the recent paper [88], where PINNs have been rigorously proven to overcome the curse of dimensionality when solving Kolmogorov equations in many dimensions. This opens up new possibilities for solving high-dimensional PDEs, such as the radiative transport equation, Boltzmann equation, and Kolmogorov equation, using PINNs.

In Chapter 6, we specifically focused on the radiative transfer equation as an illustrative example of a high-dimensional PDE, and successfully solved it using PINNs. We conducted a series of numerical experiments, ranging from the simplest monochromatic stationary radiative transfer in one space dimension (6.24) to the more complex time-dependent polychromatic radiative transfer in three space dimensions. PINNs exhibited excellent performance across all the experiments, achieving low errors with short training times. Notably, the results were qualitatively and quantitatively comparable to published findings, but with potentially significantly reduced computational costs. We supplemented the experimental results with rigorous error estimates that bounded the generalization error (6.21) in terms of computable training errors (6.22) and the number of quadrature points, independent of the underlying dimension. Refer to bounds (9.3) for detailed information. Furthermore, we also addressed an example of a parameter identification inverse problem for the radiative transfer equation using the PINN algorithm. In this problem, the objective was to compute the unknown absorption coefficient based on measurements of the incident radiation. The PINN algorithm proved to be both fast and accurate in solving this inverse problem.

Finally, we concluded the first part of the thesis centered on PINNs by considering the Kolmogorov PDE as an additional prototypical example of high-dimensional PDEs. For this equation we only provided experimental evidence of the fact that PINNs can overcome the curse of dimensionality for this equation, whereas a rigorous estimate of the total error was established in [88]. We examined different examples of the Kolmogorov PDE, including the well-known heat equation and Black-Scholes equations, across a wide range of spatial dimensions. The results demonstrated that PINNs can achieve highly accurate solutions with an error below 2%, even for 100 dimensions. Most importantly, the training time and corresponding generalization error scaled linearly with the number of dimensions, confirming the ability of PINNs to overcome the curse of dimensionality.

In the last chapter of thesis, we focused on operator learning and specifically addressed a large class of inverse problems that are only well-defined when the underlying inverse operator (2.16), maps an operator (the boundary observation operator (2.14)) to the underlying coefficient (a function). The resulting inverse problem amounts to inferring the unknown coefficient a from data pairs (Λ_a, a), with Λ_a representing the observation operator. Existing operator learning frameworks such as DeepONets (7.8) and FNOs (7.15) only map functions to functions. Hence, one needs to adapt them to be able to learn *mappings between operators and functions* in order to solve the inverse problem (2.16). To this end, we proposed a novel architecture, termed *Neural Inverse Operators* (NIO), based on a composition of DeepONets and FNOs, augmented with suitable architectural priors (definition of R in (7.38)), and trained with *randomized batching*, to guarantee invariance of the generalization error to the different discretization of the input operator. Our architecture is motivated by the underlying structure of the inverse map. We tested the NIO on a variety of benchmark inverse problems. These include the Calderón Problem in electrical impedance tomography, inverse wave scattering modelled with the Helmholtz equation, optical imaging using the radiative transport equation, and seismic imaging with the acoustic wave equation. For all these problems, NIO outperformed baselines significantly and provided accurate and, more importantly, robust approximations to the unknown coefficients with small errors. Finally, a series of experiments were also presented to demonstrate that NIO is robust with respect to various factors such as varying sensor locations, grid resolutions, noise, and discretizations of the input operator while being able to generalize *out-of-distribution* and being more accurate and much faster than direct and PDE constrained optimization algorithms. Various extensions of the architecture are possible. For instance, other architectures, such as recently proposed LOCA [40], VIDON [38], or graph-based approaches [117, 118], can be adapted in this context. Problems in higher-dimensional (particularly with seismic) imaging need to be considered to explore how NIOs scale with increasing problem size. Finally, approximation bounds and universality results, in the spirit of [119, 120] need to be derived in order to place NIOs on a solid theoretical footing.

9 Appendix

9.1 Radiative Transfer

9.1.1 Estimates on the Generalization Error for the Radiative Transfer Equation

In order to derive an error estimate for the PINNs algorithm, we need to make some assumptions on the scattering kernel Φ in (6.12). We follow standard practice and assume that it is symmetric $\Phi(\omega, \omega', \nu, \nu') = \Phi(\omega', \omega, \nu', \nu)$. Moreover, the following function,

$$\Psi(\omega, \nu) = \int\limits_{S\times\Lambda} \Phi(\omega, \omega', \nu, \nu') d\omega' d\nu', \tag{9.1}$$

is essentially bounded i.e. $\Psi \in L^\infty(S \times \Lambda)$. We have the following estimate on the generalization error (6.21),

Lemma 9.1.1. *Let $u \in L^2(D_T)$ be the unique weak solution of the radiative transfer equation* (6.12)*, with absorption coefficient $0 \le k \in L^\infty(D \times \Lambda)$, scattering coefficient $0 \le \sigma \in L^\infty(D \times \Lambda)$ and a symmetric scattering kernel $\Phi \in C^\ell(S \times \Lambda \times S \times \Lambda)$, for some $\ell \ge 1$, such that the function Ψ (defined in* (9.1)*) is in $L^\infty(S \times \Lambda)$. Let $u^* = u_{\theta^*} \in C^\ell(D_T)$ be the output of the PINNs algorithm 1 for approximating the radiative transfer equation* (6.12)*, such that*

$$\max\{V_{HK}(u^*), V_{HK}\left(r_{int,\theta^*}\right)\} < +\infty, \tag{9.2}$$

with V_{HK} being the so-called Hardy-Krause variation (see [50, 72] for the precise definition). We also assume that the initial data u_0 and boundary data u_b are of bounded Hardy-Krause variation. Then, under the assumption that Sobol points are used as the training points $\mathcal{S}_{int}, \mathcal{S}_{sb}, \mathcal{S}_{tb}$ in algorithm 1 and Guass-quadrature rule of order $s = s(\ell)$ is used in approximating the scattering kernel in the residual (3.2)*, we have the following estimate on the generalization error,*

$$\begin{aligned}(\mathcal{E}_G)^2 &\le C\left((\mathcal{E}_T^{tb})^2 + c(\mathcal{E}_T^{sb})^2 + c(\mathcal{E}_T^{int})^2\right) \\ &+ CC^*\left(\frac{(\log(N_{tb}))^{2d}}{N_{tb}} + c\frac{(\log(N_{sb}))^{2d}}{N_{sb}} + c\frac{(\log(N_{int}))^{2d+1}}{N_{int}} + cN_S^{-2s}\right)\end{aligned} \tag{9.3}$$

with constants defined as,

$$\begin{aligned} C &= T + c\hat{C}T^2 e^{c\hat{C}T}, \quad \hat{C} = 2 + \frac{2(\|\sigma\|_{L^\infty} + \|\Psi\|_{L^\infty})}{s_d} \\ C^* &= \max\left\{V_{HK}\left((r_{tb}^*)^2\right), V_{HK}\left((r_{sb}^*)^2\right), V_{HK}\left((r_{int}^*)^2\right), \overline{C}\right\} \\ \overline{C} &= \overline{C}\left(|D_T|, \|\Phi\|_{C^\ell}, \|u^*\|_{C^\ell}\right)\end{aligned} \tag{9.4}$$

Proof. We drop the θ^* dependence in the residuals (3.2), (6.19), for notational convenience and denote the residuals as $r^*_{int}, r^*_{sb}, r^*_{tb}$. Define,

$$E(u^*,\Phi) := \sum_{i=1}^{N_S} w_i^S \Phi(\omega,\omega_i^S,\nu,\nu_i^S) u^*(t,x,\omega_i^S,\nu_i^S) - \int\limits_\Lambda \int\limits_S \Phi(\omega,\omega',\nu,\nu') u^*(t,x,\omega',\nu') d\omega' d\nu'. \tag{9.5}$$

It is straightforward to derive from the radiative transfer equation (6.12) and the definition of residuals (3.2), (6.19), that the error $\hat{u} = u^* - u$, satisfies the following integro-differential equation,

$$\begin{aligned} \frac{1}{c}\hat{u}_t + \omega\cdot\nabla_x \hat{u} &= -(k+\sigma)\hat{u} + \frac{\sigma}{s_d}\int\limits_\Lambda\int\limits_S \Phi(\omega,\omega',\nu,\nu')\hat{u}(t,x,\omega',\nu')d\omega' d\nu' \\ &\quad + r^*_{int} + E(u^*,\Phi). \\ \hat{u}(0,x,\omega,\nu) &= r^*_{tb}, \quad (x,\omega,\nu)\in D\times S\times\Lambda, \\ \hat{u}(t,x,\omega,\nu) &= r^*_{sb}, \quad (t,x,\omega,\nu)\in \Gamma_-\times\Lambda. \end{aligned} \tag{9.6}$$

Multiplying $\hat{u}$ on both sides of the first equation in (9.6), we obtain,

$$\begin{aligned} \frac{1}{2c}\frac{d(\hat{u}^2)}{dt} + \omega\cdot\nabla_x(\frac{\hat{u}^2}{2}) &= -(k+\sigma)\hat{u}^2 + \frac{\sigma}{s_d}\int\limits_\Lambda\int\limits_S \Phi(\omega,\omega',\nu,\nu')\hat{u}(t,x,\omega',\nu')\hat{u}(t,x,\omega,\nu)d\omega' d\nu' \\ &\quad + r^*_{int}\hat{u} + E(u^*,\Phi)\hat{u} \end{aligned} \tag{9.7}$$

Integrating the above over $D\times S\times\nu$, integrating by parts and using the Cauchy's inequality and the fact that $k,\sigma \geq 0$, we obtain for any $t\in(0,T]$,

$$\begin{aligned} \frac{1}{2c}\frac{d}{dt}\int_{D\times S\times\Lambda}\hat{u}^2(t,x,\omega,\nu)dxd\omega d\nu &\leq \int_{D\times S\times\Lambda}\hat{u}^2(t,x,\omega,\nu)dxd\omega d\nu - \int\limits_{(\partial D\times S\times\Lambda)_-}(\omega\cdot n(x))\frac{\hat{u}^2(t,x,\omega,\nu)}{2}ds(x)d\omega d\nu \\ &\quad + \int\limits_{D\times S\times\Lambda}\frac{\sigma}{s_d}\int\limits_\Lambda\int\limits_S \Phi(\omega,\omega',\nu,\nu')\hat{u}(t,x,\omega',\nu')\hat{u}(t,x,\omega,\nu)d\omega' d\nu' d\nu d\omega dx, \\ &\quad + \int\limits_{D\times S\times\Lambda}\frac{(r^*_{int}(t,x,\omega,\nu))^2}{2}d\nu d\omega dx + \int\limits_{D\times S\times\Lambda}\frac{(E(u^*,\Phi)(t,x,\omega,\nu))^2}{2}d\nu d\omega dx \end{aligned} \tag{9.8}$$

Here $ds(x)$ denotes the surface measure on ∂D and we define

$$(\partial D\times S\times\Lambda)_- := \{(x,\omega,\nu)\in\partial D\times S\times\Lambda : \omega\cdot n(x)\leq 0\},$$

with $n(x)$ being the unit outward normal at $x\in\partial D$.

We fix any $\bar{T}\in(0,T]$ and integrate (9.8) over $(0,\bar{T})$ and estimate the result to obtain,

$$\begin{aligned} \int\limits_{D\times S\times\Lambda}\hat{u}^2(\bar{T},x,\omega,\nu)dxd\omega d\nu &\leq \int\limits_{D\times S\times\Lambda}\hat{u}^2(0,x,\omega,\nu)dxd\omega d\nu + 2c\int\limits_0^{\bar{T}}\int_{D\times S\times\Lambda}\hat{u}^2(t,x,\omega,\nu)dtdxd\omega d\nu \\ &\quad + c\int\limits_{\Gamma_-}|\omega\cdot n|\hat{u}^2(t,x,\omega,\nu)dtds(x)d\omega d\nu + I + c\int\limits_{D_T}(r^*_{int})^2 dz + c\int\limits_{D_T}(E(u^*,\Phi))^2 dz. \end{aligned} \tag{9.9}$$

Here, the term I in (9.9), is defined and estimated by successive applications of Cauchy-Schwatrz inequality as,

$$
\begin{aligned}
I &= 2c \int_0^{\bar{T}} \int_{D\times S\times\Lambda} \frac{\sigma}{s_d} \int_\Lambda \int_S \Phi(\omega,\omega',\nu,\nu')\hat{u}(t,x,\omega',\nu')\hat{u}(t,x,\omega,\nu) d\omega' d\nu' d\nu d\omega dx dt, \\
&\le \frac{2c(\|\sigma\|_{L^\infty} + \|\Psi\|_{L^\infty})}{s_d} \int_0^{\bar{T}} \int_{D\times S\times\Lambda} \hat{u}^2(t,x,\omega,\nu) dt dx d\omega d\nu.
\end{aligned}
$$

By identifying constant $\hat{C}$ from (9.4), we obtain from (9.9) and (9.6) that,

$$
\begin{aligned}
\int_{D\times S\times\Lambda} \hat{u}^2(\bar{T},x,\omega,\nu) dx d\omega d\nu \le & \int_{D\times S\times\Lambda} (r_{tb}^*)^2 dx d\omega d\nu + c\int_{\Gamma_-} (r_{sb}^*)^2 dt ds(x) d\omega d\nu \\
& + c\int_{D_T} (r_{int}^*)^2 dz + c\int_{D_T} (E(u^*,\Phi))^2 dz \\
& + c\hat{C} \int_0^{\bar{T}} \int_{D\times S\times\Lambda} \hat{u}^2(t,x,\omega,\nu) dt dx d\omega d\nu.
\end{aligned}
\tag{9.10}
$$

Applying the integral form of Grönwall's inequality to (9.10), we obtain for any $0 < \bar{T} \le T$,

$$
\begin{aligned}
\int_{D\times S\times\Lambda} \hat{u}^2(\bar{T},x,\omega,\nu) dx d\omega d\nu \le & \left(1 + c\hat{C}\bar{T}e^{c\hat{C}\bar{T}}\right) \left(\int_{D\times S\times\Lambda} (r_{tb}^*)^2 dx d\omega d\nu + c\int_{\Gamma_-} (r_{sb}^*)^2 dt ds(x) d\omega d\nu \right) \\
& + \left(1 + c\hat{C}\bar{T}e^{c\hat{C}\bar{T}}\right) \left(c\int_{D_T} (r_{int}^*)^2 dz + c\int_{D_T} (E(u^*,\Phi))^2 dz \right)
\end{aligned}
\tag{9.11}
$$

Integrating (9.11) over $(0,T)$ yields,

$$
\begin{aligned}
(\mathcal{E}_G)^2 := \int_{D_T} \hat{u}^2(t,x,\omega,\nu) dz \le & \left(T + c\hat{C}T^2 e^{c\hat{C}T}\right) \left(\int_{D\times S\times\Lambda} (r_{tb}^*)^2 dx d\omega d\nu + c\int_{\Gamma_-} (r_{sb}^*)^2 dt ds(x) d\omega d\nu \right) \\
& + \left(T + c\hat{C}T^2 e^{c\hat{C}T}\right) \left(c\int_{D_T} (r_{int}^*)^2 dz + c\int_{D_T} (E(u^*,\Phi))^2 dz \right)
\end{aligned}
\tag{9.12}
$$

As the training points in $\mathcal{S}_{tb}$ are the Sobol quadrature points, we realize that the training error $(\mathcal{E}_T^{tb})^2$ (6.22) is the quasi-Monte Carlo quadrature for the first integral in (9.12). Hence by the well-known Koksma-Hlawka inequality [50], we obtain the following estimate,

$$
\int_{D\times S\times\Lambda} (r_{tb}^*)^2 dx d\omega d\nu \le (\mathcal{E}_T^{tb})^2 + V_{HK}\left((r_{tb}^*)^2\right) \frac{(\log(N_{tb}))^{2d}}{N_{tb}}. \tag{9.13}
$$

By a similar argument, we can estimate,

$$
\begin{aligned}
\int\limits_{\Gamma_-} (r_{sb}^*)^2 dt ds(x) d\omega d\nu \leq (\mathcal{E}_T^{sb})^2 + V_{HK}\left((r_{sb}^*)^2\right) \frac{(\log(N_{sb}))^{2d}}{N_{sb}}, \\
\int\limits_{D_T} (r_{int}^*)^2 dz \leq (\mathcal{E}_T^{int})^2 + V_{HK}\left((r_{int}^*)^2\right) \frac{(\log(N_{int}))^{2d+1}}{N_{int}},
\end{aligned}
\tag{9.14}
$$

As ω_i^S, ν_i^S, for $1 \leq i \leq N_S$ are Gauss-quadrature points, we follow [49] and readily estimate E defined in (9.5) by the error for an s-th order accurate Gauss quadrature rule with $s = s(\ell)$ as,

$$
\int\limits_{D_T} (E(u^*, \Phi))^2 dz \leq \overline{C} N_S^{-2s}, \tag{9.15}
$$

with constant $\overline{C}$ defined in (9.4) By plugging in the estimates (9.13), (9.14), (9.15) in (9.12) and identifying constants, we derive the desired estimate (9.3) on the generalization error (6.21). ☐

9.1.2 Estimates on the Generalization Error in the Steady Case

The steady-state (time-independent) version of the radiative transfer equation (6.12) is obtained by letting the speed of light $c \to \infty$ and resulting in,

$$
(k+\sigma)u = -\omega \cdot \nabla_x u + \frac{\sigma}{s_d} \int\limits_{\Lambda} \int\limits_{S} \Phi(\omega, \omega', \nu, \nu') u(x, \omega', \nu') d\omega' d\nu' + f, \tag{9.16}
$$

with all the coefficients and sources as defined before. We also impose the *inflow* boundary condition,

$$
u(x, \omega, \nu) = u_b(x, \omega, \nu), \quad (x, \omega, \nu) \in \Gamma^s_, \tag{9.17}
$$

with inflow boundary defined by,

$$
\Gamma_-^s = \{(x, \omega, \nu) \in \partial D \times S \times \Lambda : \omega \cdot n(x) < 0\} \tag{9.18}
$$

with $n(x)$ denoting the unit outward normal at any point $x \in \partial D$.

The PINNs algorithm 1 can be readily adpated to this case by simply (formally) neglecting the temporal dependence in the residuals (3.2), (6.19) and loss functions and the underlying definitions of neural networks. We omit detailing this procedure here. Our objective is to bound the resulting generalization error,

$$
\mathcal{E}_G^s = \mathcal{E}_G^s(\theta^*) := \left(\int\limits_{D \times S \times \Lambda} |u(x, \omega, \nu) - u^*(x, \omega, \nu)|^2 dz \right)^{\frac{1}{2}}, \tag{9.19}
$$

with $dz = dx d\omega d\nu$ denoting the underlying volume measure. As in lemma 9.1.1, we will bound the generalization error in terms of the training errors,

$$
\mathcal{E}_T^{sb} := \left(\sum_{j=1}^{N_{sb}} w_j^{sb} |r_{sb,\theta^*}(z_j^{sb})|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_T^{int} := \left(\sum_{j=1}^{N_{int}} w_j^{int} |r_{int,\theta^*}(z_j^{int})|^2 \right)^{\frac{1}{2}} \tag{9.20}
$$

Here, z_j^{int} and z_j^{sb} are the interior and spatial boundary training points.

We have the following estimate on the generalization error,

Lemma 9.1.2. *Let $u \in L^2(D\times S\times\Lambda)$ be the unique weak solution of the radiative transfer equation* (9.16)*, with absorption coefficient $0 < k_{min} \le k(x,\nu) \le k_{max} < \infty$, scattering coefficient $0 < \sigma_{min} \le \sigma(x,\nu) \le \sigma_{max} < \infty$, for almost every $x \in D, \nu \in \Lambda$ and a symmetric scattering kernel $\Phi \in C^\ell(S\times\Lambda\times S\times\Lambda)$, for some $\ell \ge 1$, such that the function Ψ (defined in* (9.1)*) is in $L^\infty(S\times\Lambda)$. We further assume that the absorption and scattering coefficients are related in the following manner, there exists a $\kappa > 0$, such that*

$$k_{min} + \sigma_{min} - \frac{\sigma_{max} + \|\Psi\|_{L^\infty}}{s_d} \ge \kappa \tag{9.21}$$

Let $u^* = u_{\theta^*} \in C^\ell(D\times S\times\Lambda)$ be the output of the PINNs algorithm 1 for approximating the stationary radiative transfer equation (9.16)*, such that*

$$\max\{V_{HK}(u^*), V_{HK}\left(r_{int,\theta^*}\right)\} < +\infty, \tag{9.22}$$

with V_{HK} being the Hardy-Krause variation. We also assume that the boundary data u_b is of bounded Hardy-Krause variation. Then, under the assumption that Sobol points are used as the training points $\mathcal{S}_{int}, \mathcal{S}_{sb}$ in algorithm 1 and Guass-quadrature rule of order $s = s(\ell)$ is used in approximating the scattering kernel in the residual (3.2)*, we have the following estimate on the generalization error,*

$$(\mathcal{E}_G^s)^2 \le C\left((\mathcal{E}_T^{sb})^2 + (\mathcal{E}_T^{int})^2 + \frac{(\log(N_{sb}))^{2d-1}}{N_{sb}} + \frac{(\log(N_{int}))^{2d}}{N_{int}} + N_S^{-2s}\right) \tag{9.23}$$

with constants defined as,

$$C = \max\left\{\frac{2}{\kappa}, \frac{2}{\kappa}V_{HK}\left((r_{sb}^*)^2\right), \frac{2C_\epsilon}{\kappa}\left((r_{int}^*)^2\right), \frac{2C_\epsilon}{\kappa}\overline{C}N_S^{-2s}\right\}, \tag{9.24}$$

where $\overline{C}$ is defined in (9.4)*. Here, C_ϵ is a constant that depends on κ and is defined in* (9.29)*.*

Proof. We drop the θ^* dependence in the residuals (3.2), (6.19), for notational convenience and denote the residuals as r_{int}^*, r_{sb}^*. Define,

$$E_s(u^*,\Phi) := \sum_{i=1}^{N_S} w_i^S \Phi(\omega,\omega_i^S,\nu,\nu_i^S)u^*(x,\omega_i^S,\nu_i^S) - \int\limits_\Lambda\int\limits_S \Phi(\omega,\omega',\nu,\nu')u^*(x,\omega',\nu')d\omega' d\nu'. \tag{9.25}$$

It is straightforward to derive from the radiative transfer equation (9.16) and the definition of residuals (3.2), (6.19), that the error $\hat{u} = u^* - u$, satisfies the following integro-differential equation,

$$\begin{aligned}(k+\sigma)\hat{u} &= -\omega\cdot\nabla_x\hat{u} + \frac{\sigma}{s_d}\int\limits_\Lambda\int\limits_S \Phi(\omega,\omega',\nu,\nu')\hat{u}(x,\omega',\nu')d\omega' d\nu' + r_{int}^* + E_s(u^*,\Phi),\\ \hat{u}(x,\omega,\nu) &= r_{sb}^*, \quad (x,\omega,\nu)\in\Gamma_-^s\end{aligned} \tag{9.26}$$

Multiplying $\hat{u}$ on both sides of the first equation in (9.26), we obtain,

$$\begin{aligned}(k+\sigma)\hat{u}^2 = &-\omega\cdot\nabla_x(\frac{\hat{u}^2}{2}) + \frac{\sigma}{s_d}\int\limits_\Lambda\int\limits_S \Phi(\omega,\omega',\nu,\nu')\hat{u}(x,\omega',\nu')\hat{u}(x,\omega,\nu)d\omega' d\nu'\\ &+ r_{int}^*\hat{u} + E_s(u^*,\Phi)\hat{u}\end{aligned} \tag{9.27}$$

Integrating the above over $D \times S \times \nu$, integrating by parts, using the assumed lower and upper bounds on k, σ, we obtain,

$$(k_{min} + \sigma_{min}) \int\limits_{D\times S\times\nu} \hat{u}^2 dz \leq \int\limits_{\Gamma^s_-} (r^*_{sb})^2 ds(x) d\omega d\nu + I + \int\limits_{D\times S\times\nu} (r^*_{int}\hat{u} + E_s(u^*, \Phi)\hat{u}) dz, \tag{9.28}$$

with term I defined and estimated by,

$$\begin{aligned} I &= \int\limits_{D\times S\times\Lambda} \frac{\sigma}{s_d} \int\limits_{\Lambda}\int\limits_{S} \Phi(\omega, \omega', \nu, \nu')\hat{u}(x, \omega', \nu')\hat{u}(x, \omega, \nu) d\omega' d\nu' d\nu d\omega dx, \\ &\leq \frac{\sigma_{max} + \|\Psi\|_{L^\infty}}{s_d} \int_{D\times S\times\Lambda} \hat{u}^2(x, \omega, \nu) dz. \end{aligned}$$

From the assumption (9.21), there exists an $\epsilon > 0$ such that $k_{min} + \sigma_{min} - \frac{\sigma_{max} + \|\Psi\|_{L^\infty}}{s_d} - 2\epsilon > \frac{\kappa}{2}$, we use the ϵ-version of Cauchy's inequality,

$$ab \leq \epsilon a^2 + C_\epsilon b^2, \tag{9.29}$$

to further estimate (9.28) as,

$$\int\limits_{D\times S\times\Lambda} \hat{u}^2 dz \leq \frac{2}{\kappa} \int\limits_{\Gamma^s_-} (r^*_{sb})^2 ds(x) d\omega d\nu + \frac{2C_\epsilon}{\kappa} \left(\int\limits_{D\times S\times\nu} (r^*_{int})^2 + (E_s(u^*, \Phi))^2 dz \right) \tag{9.30}$$

By using the estimates (9.14) and (9.15) and identifying constants, we obtain the desired bound (9.23) on the generalization error (9.19).

□

As for the time-dependent case, the bound (9.23) should be considered in the sense of *if the PINN is trained well, it generalizes well.* Moreover, the bound, and consequently, the PINN does not suffer from a curse of dimensionality by the same argument as in the time-dependent case. Infact, the logarithmic corrections to the linear decay of the rhs in (9.23) can be ignored at an even smaller number of training points.

The assumption (9.21) plays a key role in the derivation of the bound (9.23). A careful inspection of this assumption reveals that the scattering coefficient is not allowed to vary over a large range, unless there is enough absorption in the medium. However, there is no restriction on the range of scales over which the absorption coefficient can vary.

9.2 Neural Inverse Operator

9.2.1 Architecture Details

Feed Forward Dense Neural Networks

A description of feed-forward dense neural network is provided in 2.5.1 In all numerical experiments, the trunk net of DeepONet is a feed-forward neural network. We consider a uniform number of neurons across all the layers of the network $d_\ell = d_{\ell-1} = d$, $1 < \ell < L_t$.

Fully Convolutional Neural Network

Fully convolutional neural networks are a special class of convolutional networks which can be evaluated for virtually any resolution of the input. A detailed description of CNN and FCNN is provided in Section 2.5.2.

A visual representation of the convolutional architectures used for the benchmark problems is depicted in Figures 9.1, 9.2, 9.3,9.4. The *convolutional block* (or *transposed convolution block*) is the composition of a convolution (or transposed convolution) operation, batch normalization, and activation function (Leaky ReLU). The cropping operation involves adding negative padding to the edges of a tensor to achieve the desired output width and height. The number of channels c is selected with cross-validation. The architecture used for seismic imagining is referred to as InversionNet in [110].

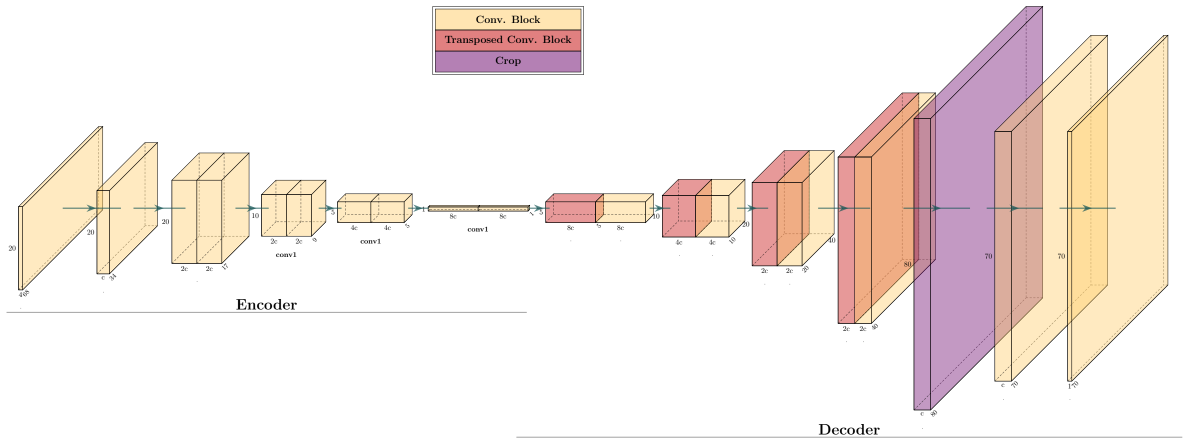


Figure 9.1: Schematic representation of the Fully-Convolutional Neural Network (FCNN) architecture used for the Calderón problem with Trigonometric coefficients and for the Inverse Wave Scattering with Helmholtz Equation.

DeepONet

The architectures of the branch and trunk are chosen according to the benchmark addressed. In particular, we employ standard feed-forward neural networks as trunk-net in all the experiments. In contrast, the branch is obtained as a composition of the *encoder* of the fully convolutional networks depicted in figures 9.1, 9.2 9.3 and 9.4, and a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^p$, where n denotes the number of channels in the last layer of the encoder and p the number of basis functions. Moreover, $c = 32$ for the seismic imaging and $c = 64$ for all the other benchmarks.

Hence, the architecture of the branch is fixed. The number of the trunk hidden layers L_t, units d, and p are chosen through cross-validation. On the other hand, the activation function σ is chosen to be a leaky ReLU for both the branch and the trunk.

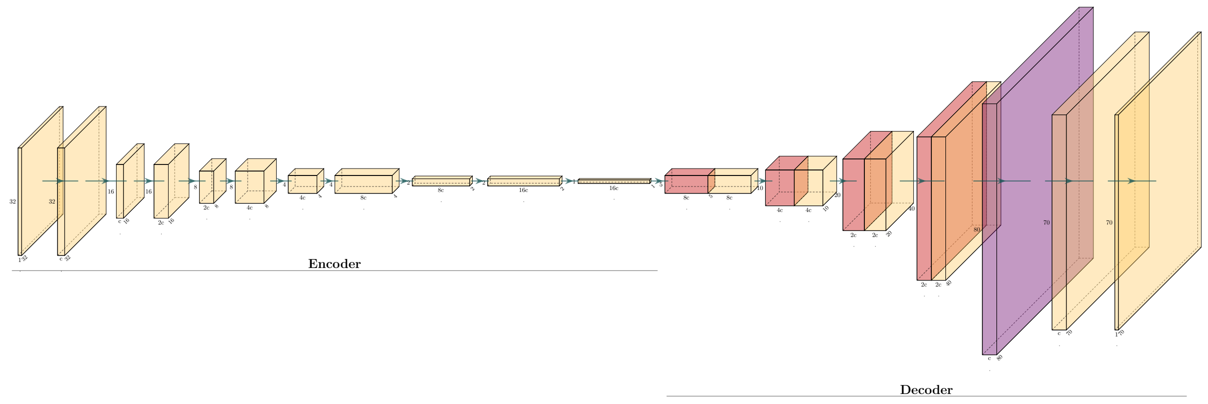


Figure 9.2: Schematic representation of the Fully-Convolutional Neural Network architecture used for the optical imaging for the Calderón Problem with Heart&Lungs phantom.

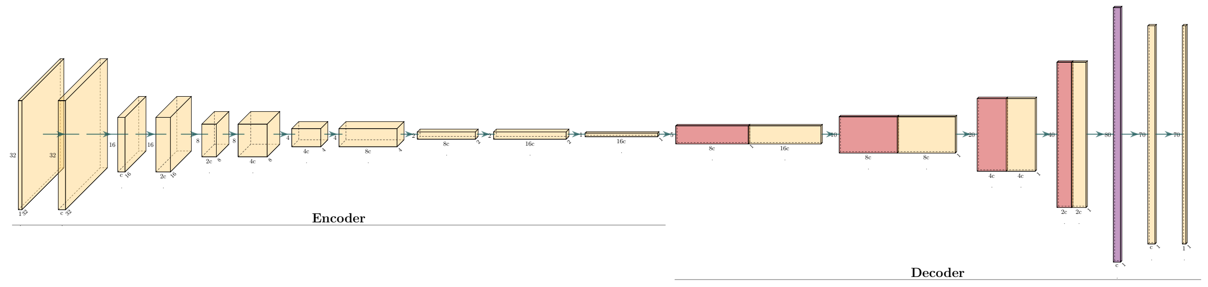


Figure 9.3: Schematic representation of the Fully-Convolutional Neural Network architecture used for the optical imaging for the radiative transport Equation.

Fourier Neural Operator

We use the implementation of the FNO model provided by the authors of [35]. Specifically, the projection Q to the target space is performed by a shallow neural network with a single hidden layer with 128 neurons and *GeLU* activation function. The same activation function is also used for all the Fourier layers. Moreover, $b_\ell(x) = 0$, for all $\ell = 1, \dots, T$ and the weight matrix W_ℓ used in the residual connection derives from a convolutional layer defined by $(k_\ell = 1, s = 1, p = 0, c_\ell = d_v, c_{\ell+1} = d_v)$, for all $1 < \ell < T$.

Neural Inverse Operator

In all numerical experiments, the proposed architecture is constructed by combining the DeepONet and Fourier Neural Operator by means of the lifting operator R defined in equation (7.38).

The implementation of DeepONet follows the same description as outlined in Section 9.2.1. However, the branch structure differs from the *encoder* structure depicted in figures 9.1, 9.2, 9.3, and 9.4. In the proposed NIO architecture, the channel mixing is performed downstream through the operator $\mathcal{M}$.

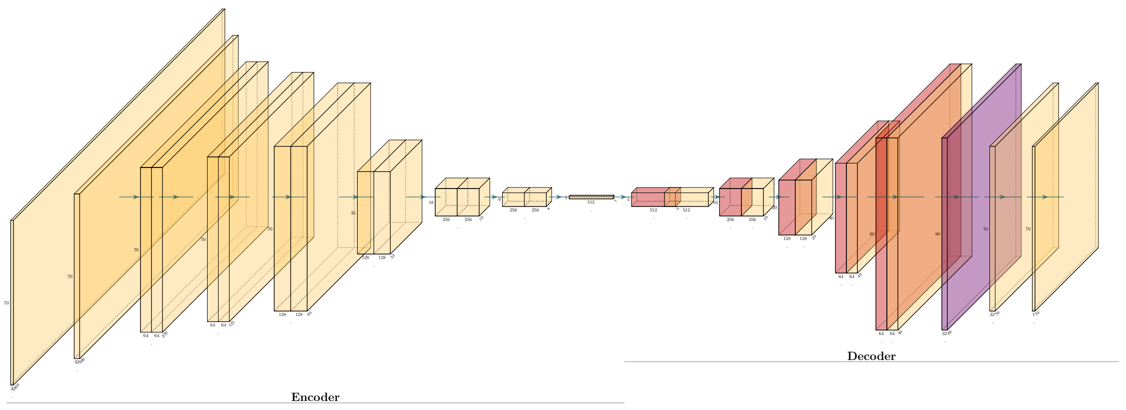


Figure 9.4: Schematic representation of the Fully-Convolutional Neural Network architecture used for the seismic imaging problems.

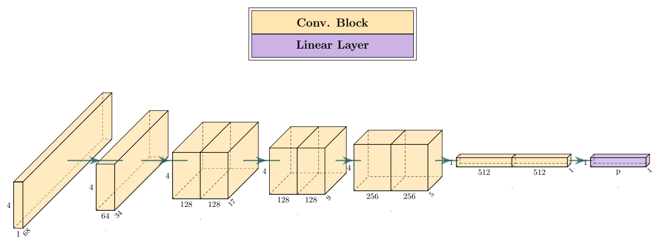


Figure 9.5: Schematic representation of the NIO-BranchNet architecture used for the Calderón problem with Trigononmetric coefficients and for the Inverse Wave Scattering with Helmholtz Equation

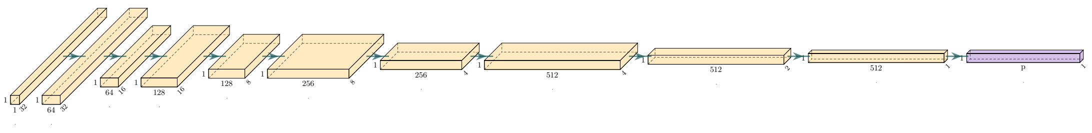


Figure 9.6: Schematic representation of the NIO-BranchNet architecture used for the optical Imaging with Radiative transport Equation and Calderón Problem with Heart&Lungs phantom.

Specific details regarding the branch-net architectures used in NIO for the benchmark problems can be found in Figures 9.5, 9.6, and 9.7. For instance, for the seismic imaging, the branch architecture is the same as the *encoder* shown in Figure 9.4, but only a single input channel is used instead of five, following the same rationale as mentioned above. Overall, the model includes the following hyperparameters: the number of layers L_t and neurons d of the DeepONet trunk, the number of basis functions p, and the lifting dimension d_v, the number of Fourier layers T and number of (truncated)-Fourier coefficients k, of FNO.

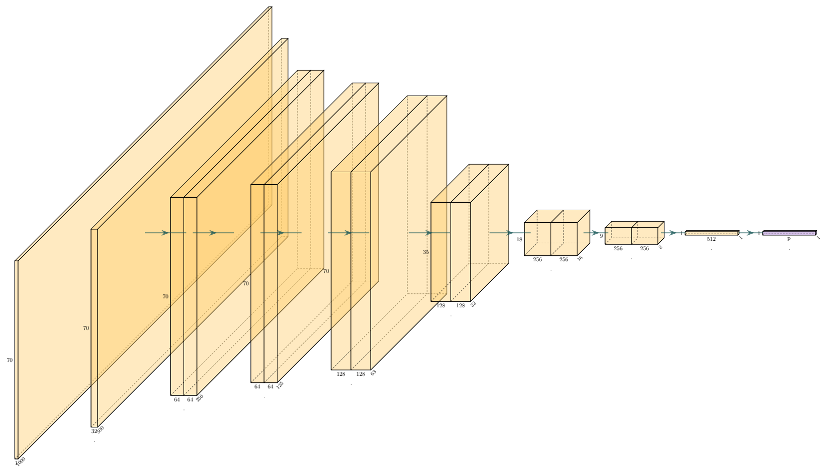


Figure 9.7: Schematic representation of the NIO-BranchNet architecture used for the seismic imaging.

9.2.2 Training Details

The training of the models, including the baselines, is performed with the ADAM optimizer, with a learning rate η for 1000 epochs (250 epochs in the Seismic imaging problem) and minimizing the L^1-loss function. We also use a step learning rate scheduler and reduce the learning rate of each parameter group by a factor γ every epoch. We train the models in mini-batches of size 256, and a weight decay of magnitude w is used. Moreover, the input and output data are transformed with a suitable map before training. Observe that the testing error reported in Table 4.4 has been obtained on the non-transformed output data. We consider two different data transformations to preprocess the data:

1. *MinMax*. This transformation involves scaling both the inputs and outputs to a range between -1 and 1:
$$\tilde{f} = 2\frac{f-m}{M-m} - 1, \tag{9.31}$$
where M and m are the maximum and the minimum value of f across all the *training samples.*

2. *log-MinMax*. This transformation is specifically used for Seismic Imaging problems. The input data are transformed according to the following equation:
$$\tilde{f} = \log\left(|f|\right)\operatorname{sign}\left(f\right), \tag{9.32}$$
and then, the obtained input and output scaled between -1 and 1.

All the parameters mentioned above, including the type of data transformation (*Identity*, *MinMax*, *log-MinMax*, are chosen through cross-validation.

At every epoch, the relative L^1 error is computed on the validation set, and the set of trainable parameters resulting in the lowest error during the entire process is saved for testing. Early stopping is used to interrupt the training if the best validation error does not improve after 50 epochs.

The cross-validation is performed by running a random search over a chosen range of hyperparameters values and selecting the configuration, realizing the lowest relative L^1 error on the validation set. Overall, 50 hyperparameter configurations are tested for NIO and 30 for the baselines. The model size (minimum and maximum number of trainable parameters) covered in this search are reported in Table 9.1.

The results of the random search, i.e., the best-performing hyperparameter configurations for each model and each benchmark, are reported in tables 9.2, 9.3, and 9.4. The FCNN hyperparameters reported in the table for the seismic imaging problem are those used in [110].

	Calderón Problem Trigonometric	**Calderón Problem Heart&Lungs**	**Inverse Wave Scattering**	**Radiative Transport**	**Seismic Imaging CurveVel - A**	**Seismic Imaging Style - A**
DONet	4.6M 9.07M	4.6M 9.07M	9.54M 14.01M	9.54M 14.01M	12.01M 15.85M	12.01M 15.85M
FCNN	1.07M 68.32M	1.07M 68.32M	4.31M 275.37M	2.48M 39.53M	24.4M 24.4M	24.4M 24.4M
NIO	7.95M 27.79M	10.6M 50.76M	7.95M 27.79M	9.6M 10.3M	13.07M 32.91M	13.07M 32.91M

Table 9.1: Minimum (Top sub-row) and maximum (Bottom sub-row) number of trainable parameters among the random-search hyperparameters configurations for all the models in every problem reported in Table 4.4 in the main text.

	η	γ	w	Data Trans	p	L_t	d	Trainable Params
Calderón Problem Trigonometric	0.001	1.0	0.0	Identity	25	8	200	4.84M
Calderón Problem Heart&Lungs	0.001	1.0	0.0	MinMax	100	15	500	13.1M
Inverse Wave Scattering	0.001	1.0	1e-06	Identity	1000	12	500	8.32M
Radiative transport	0.001	1.0	0.0	MinMax	100	15	500	13.1M
Seismic Imaging CurveVel - A	0.001	0.98	1e-06	log-MinMax	400	12	500	15.09M
Seismic Imaging Style - A	0.001	0.98	1e-06	log-MinMax	400	12	500	15.09M

Table 9.2: DeepONet best-performing hyperparameters configuration for different benchmark problems.

	η	γ	w	Data Trans	c	Trainable Params
Calderón Problem Trigonometric	0.001	1.0	0.0	MinMax	16	1.07M
Calderón Problem Heart&Lungs	0.001	1.0	0.0	MinMax	128	275.37M
Inverse Wave Scattering	0.001	1.0	0.0	MinMax	128	68.32M
Radiative transport	0.001	1.0	1e-06	MinMax	16	2.48M
Seismic Imaging CurveVel - A	0.001	1	1e-04	log-MinMax	64	24.4M
Seismic Imaging Style - A	0.001	1	1e-04	log-MinMax	64	24.4M

Table 9.3: Fully convolutional neural network best-performing hyperparameters configuration for different benchmark problems.

	η	γ	w	Data Trans	p	L_t	d	k	d_v	L	Trainable Params
Calderón Problem Trigonometric	0.001	1.0	1e-06	Identity	100	8	100	25	32	4	12.06M
Calderón Problem Heart&Lungs	0.001	1.0	1e-06	MinMax	100	8	100	25	32	4	14.74M
Inverse Wave Scattering	0.001	0.98	0.0	Identity	100	8	200	16	64	4	15.57M
Radiative transport	0.001	0.98	1e-06	MinMax	400	4	100	32	64	4	10.3M
Seismic Imaging CurveVel - A	0.001	0.98	1e-06	MinMax	25	4	200	16	32	3	16.49M
Seismic Imaging Style - A	0.001	0.98	1e-06	log-MinMax	100	8	200	16	64	2	16.49M

Table 9.4: Neural Inverse Operator best-performing hyperparameters configuration for different benchmark problems.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[2] R Evans, J Jumper, J Kirkpatrick, L Sifre, T Green, C Qin, A Zidek, A Nelson, A Bridgland, H Penedones, et al. De novo structure prediction with deep-learning based scoring. *Annual Review of Biochemistry*, 77(6):363–382, 2018.

[3] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[4] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[5] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen. Solving stochastic differential equations and kolmogorov equations by means of deep learning. Preprint, available as arXiv:1806.00421v1.

[6] F. Laakmann and P. Petersen. Efficient approximation of solutions of parametric linear transport equations by reludnns. Preprint, 2019.

[7] Siddhartha Mishra. A machine learning framework for data driven acceleration of computations of differential equations. *Mathematics in Engineering*, 1:118, 2019.

[8] Deep Ray and Jan S Hesthaven. An artificial neural network as a troubled-cell indicator. *Journal of Computational Physics*, 367:166–191, 2018.

[9] Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6111–6122. Curran Associates, Inc., 2020.

[10] Christoph Schwab and Jakob Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq. *Analysis and Applications*, 17(01):19–55, 2019.

[11] Gitta Kutyniok, Philipp Petersen, Mones Raslan, and Reinhold Schneider. A theoretical analysis of deep neural networks and parametric pdes. *Constructive Approximation*, pages 1–53, 2021.

[12] Kjetil O Lye, Siddhartha Mishra, and Deep Ray. Deep learning observables in computational fluid dynamics. *Journal of Computational Physics*, page 109339, 2020.

[13] Kjetil O Lye, Siddhartha Mishra, Deep Ray, and Praveen Chandrashekar. Iterative surrogate model optimization (ISMO): An active learning algorithm for pde constrained optimization with deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 374:113575, 2021.

[14] G. Cybenko. Approximations by superpositions of sigmoidal functions. *Approximation theory and its applications.*, 9(3):17–28, 1989.

[15] D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

[16] Kjetil O Lye, Siddhartha Mishra, and Roberto Molinaro. A multi-level procedure for enhancing accuracy of machine learning algorithms. *European Journal of Applied Mathematics*, 2020.

[17] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data. *arXiv preprint arXiv:1808.04327*, 2018.

[18] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. Deepxde: A deep learning library for solving differential equations. Preprint, available from arXiv:1907.04502, 2019.

[19] Z. Mao, A. D. Jagtap, and G. E. Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

[20] G. Pang, L. Lu, and G. E. Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM journal of Scientific computing*, 41:A2603–A2626, 2019.

[21] Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Finite basis physics-informed neural networks (fbpinns): a scalable domain decomposition approach for solving differential equations, 2021.

[22] Y. Shin, J. Darbon, and G. E. Karniadakis. On the convergence and generalization of physics informed neural networks. Preprint, available from arXiv:2004.01806v1, 2020.

[23] Helge Holden and Nils Henrik Risebro. *Front tracking for hyperbolic conservation laws*, volume 152. Springer, 2015.

[24] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[25] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. Preprint, available from arXiv:1912.01085, 2019.

[26] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[27] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[28] Z. Mao, L. Lu, O. Marxen, T. Zaki, and G. E. Karniadakis. DeepMandMnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. Preprint, available from arXiv:2011.03349v1, 2020.

[29] Shengze Cai, Zhicheng Wang, Lu Lu, Tamer A Zaki, and George Em Karniadakis. Deepm&mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436:110296, 2021.

[30] Samuel Lanthaler, Roberto Molinaro, Patrik Hadorn, and Siddhartha Mishra. Nonlinear reconstruction for operator learning of PDEs with discontinuities. In *The Eleventh International Conference on Learning Representations*, 2023.

[31] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model Reduction And Neural Networks For Parametric PDEs. *The SMAI journal of computational mathematics*, 7:121–157, 2021.

[32] N. Kovachki, Z. Li, B. Liu, K. Azizzadensheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481v3*, 2021.

[33] Zongyi Li, Nikola B Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *CoRR*, abs/2003.03485, 2020.

[34] Zongyi Li, Nikola B Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew M Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6755–6766. Curran Associates, Inc., 2020.

[35] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[36] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

[37] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, p. Hassanzadeh, K. Kashinath, and A. Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[38] M. Prasthofer, T. De Ryck, and S. Mishra. Variable input deep operator networks. *arXiv preprint arXiv:2205.11404*, 2022.

[39] V. Fanaskov and I. Oseledets. Spectral neural operators, 2022.

[40] Georgios Kissas, Jacob H Seidman, Leonardo Ferreira Guilhoto, Victor M Preciado, George J Pappas, and Paris Perdikaris. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.

[41] Jacob H Seidman, Georgios Kissas, Paris Perdikaris, and George J Pappas. NOMAD: Nonlinear manifold decoders for operator learning. *arXiv preprint arXiv:2206.03551*, 2022.

[42] Shuhao Cao. Choose a transformer: Fourier or galerkin. In *35th conference on neural information processing systems*, 2021.

[43] Bogdan RaoniÄ‡, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de BÃ©zenac. Convolutional neural operators for robust and accurate learning of pdes, 2023.

[44] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations, 2023.

[45] Gunther Uhlmann. Electrical impedance tomography and Calderón's problem. *Inverse problems*, 25(12):123011, 2009.

[46] Ru-Yu Lai, Qin Li, and Gunther Uhlmann. Inverse problems for the stationary transport equation in the diffusion scaling. *SIAM Journal on Applied Mathematics*, 79(6):2340–2358, 2019.

[47] Oz Yilmaz. *Seismic Data Analysis.* Society for exploration geophysicists, 2011.

[48] Victor Isakov. *Inverse Problems for Partial Differential Equations.* Springer, 2017.

[49] J. Stoer and R. Bulirsch. *Introduction to numerical analysis.* Springer Verlag, 2002.

[50] Russel E Caflisch. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, 7:1–49, 1998.

[51] Art B Owen. Multidimensional variation for quasi-monte carlo. In *Contemporary Multivariate Analysis And Design Of Experiments: In Celebration of Professor Kai-Tai Fang's 65th Birthday*, pages 49–74. World Scientific, 2005.

[52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[53] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[55] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[56] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

[57] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge University Press, 2014.

[58] A. Friedman. *Partial differential equations of the parabolic type.* prentice hall, 1964.

[59] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[60] E. Godlewski and P. A. Raviart. *Hyperbolic systems of conservation laws.* Ellipsis, 1991.

[61] Andrew J. Majda and Andrea L. Bertozzi. *Vorticity and Incompressible Flow.* Cambridge Texts in Applied Mathematics. Cambridge University Press, 2001.

[62] J. K Hunter and J Scheurle. Existence of perturbed solitary wave solutions to a model equation for water waves. *Physica D.*, 32:253–268, 1988.

[63] Andrei Faminskii and Nikolai Larkin. Initial-boundary value problems for quasilinear dispersive equations posed on a bounded interval. *Electronic Journal of Differential Equations*, 01:1–20, 2010.

[64] Genming Bai, Ujjwal Koley, Siddhartha Mishra, and Roberto Molinaro. Physics informed neural networks (pinns) for approximating nonlinear dispersive pdes. *Journal of Computational Mathematics*, 39(6), 2021.

[65] Jonathan T Barron. Continuously differentiable exponential linear units. *arXiv preprint arXiv:1704.07483*, 2017.

[66] J.B. Bell, P. Collela, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85:257–283, 1989.

[67] Juan Carlos Ceballos, Mauricio SepÃºlveda, and Octavio Paulo Vera VillagrÃ¡n. The kortewegâ€“de vriesâ€“kawahara equation in a bounded domain and some numerical results. *Applied Mathematics and Computation*, 190(1):912 – 936, 2007.

[68] Ujjwal Koley. Error estimate for a fully discrete spectral scheme for korteweg-de vries-kawahara equation. *Cent. Eur. J. Math.*, 10(1):173–187, 2012.

[69] Ujjwal Koley. Finite difference schemes for the kortewegâ€“de vriesâ€“kawahara equation. *Int. J. Numer. Anal. Model.*, 13(3):344–367, 2016.

[70] E Kharazmi, Z Zhang, and G. Em Karniadakis. Variational physics informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*, 2019.

[71] Tim De Ryck, Siddhartha Mishra, and Roberto Molinaro. wpinns: Weak physics informed neural networks for approximating entropy solutions of hyperbolic conservation laws. *arXiv preprint arXiv:2207.08483*, 2022.

[72] S. Mishra and T. Konstantin Rusch. Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. Preprint, available as arXiv:2005.12564, 2020.

[73] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA Journal of Numerical Analysis*, 43(1):1–43, 01 2022.

[74] G. Alessandrini, L. Rondi, E. Rosset, and S. Vessella. The stability of the cauchy problem for elliptic equations. *Inverse problems*, 25(12):47, 2009.

[75] E. Burman and L. Oksanen. Weakly consistent regularization methods for ill-posed problems. In *Numerical methods for PDEs, D. A. Di Pietro (eds)*, pages 171–202. World Scientific, 2018.

[76] O. Yu. Imanuvilov. Controllability of parabolic equations. *Math. Sb.*, 186(6):109–132, 1995.

[77] P. Kuchment and L. Kunyansky. Mathematics of thermoacoustic tomography. *European J. Appl. Math.*, 2(19):191–224, 2008.

[78] K. Ramdani, M. Tucsnak, and L. Kunyansky. Recovering the initial state of an infinite-dimennsional system using observers. *Automatica J.*, 46(10):1616–1625, 2010.

[79] C. Bardos, G. Lebeau, and J. Rauch. Un example dutilization des notions de propagation pour le controle et al stabilisation de prolémes hyperboliques. *Rend. Sem. Math. Univ. Politec. Torino*, pages 11–31, 1989.

[80] J. Le Rousseau, G. Lebeau, P. Terpolilli, and E. Trélat. Geometric control condition for the wave equation with a time-dependent observation domain. *Analysis PDE.*, 10(4):983–1015, 2017.

[81] L. Miller. Escape function conditions for the observation, control and stabilization of the wave equation. *SIAM J. Control. Opt.*, 41(5):1554–1566, 2002.

[82] E. Burman, A. Feizmohammadi, and L. Oksanen. A finite element data assimilation method for the wave equation. *Math. Comp.*, 89(324):1681–1709, 2020.

[83] L. Baudouin and M. De Buhan annd S. Ervedoza. Global carleman estimates for waves and applications. *Commun. PDE.*, 38:556–598, 2013.

[84] E. Burman and P. Hansbo. Stabilized nonconfirming finite element methods for data assimilation in incompressible flows. *Math. Comp.*, 87(311):1029–1050, 2018.

[85] C-L. Lin, G. Uhlmann, and J-N. Wang. Optimal three-ball inequalities and quantitative uniqueness for the stokes system. *Discrete Contin. Dyn. Syst.*, 28(3):1273–1290, 2010.

[86] G. Seregin. *Lecture notes on regularity theory for the Navier-Stokes equations.* World Scientific, 2015.

[87] E. Burman, J. Ish-Horowicz, and L. Oksanen. Fully discrete finite element data assimilation method for the heat equation. *ESIAM: Math. Model. Num. Anal.*, 52:2065–2082, 2018.

[88] Tim De Ryck and Siddhartha Mishra. Error analysis for physics-informed neural networks (pinns) approximating kolmogorov pdes. *Advances in Computational Mathematics*, 48(6):1–40, 2022.

[89] G. Kanschat and et. al. *Numerical methods in multi-dimensional radiative transfer.* Springer, 2008.

[90] D. Kaushik, M. Smith, A. Wollaber, B. Smith, A. Siegel, and W. S. Yang. Enabling high fidelity neutron transport simulations on petascle architectures. In *Proceedings of the Conference on High Performance Computing Networking, Storage, and Analysis, volume 67, Portland, Oregon*, 2009.

[91] M. F. Modest. *Radiative heat transfer.* Elsevier, 2003.

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

[93] M. Frank. Approximate models for radiative transfer. *Bull. Inst. Math. Acad. Sinica (New Series)*, 2:409–432, 2007.

[94] JP Pontaza and JN Reddy. Least-squares finite element formulations for one-dimensional radiative transfer. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 95(3):387–406, 2005.

[95] YA Cengel, MN Özi, et al. Radiation transfer in an anisotropically scattering plane-parallel medium with space-dependent albedo ω(x). *Journal of Quantitative Spectroscopy and Radiative Transfer*, 34(3):263–270, 1985.

[96] Konstantin Grella. *Sparse tensor approximation for radiative transport.* PhD thesis, ETH Zurich, 2013.

[97] Sabine Richling, Erik Meinköhn, N Kryzhevoi, and Guido Kanschat. Radiative transfer with finite elements-i. basic method and tests. *Astronomy & Astrophysics*, 380(2):776–788, 2001.

[98] F. Graziani. The prompt spectrum of a radiating sphere: Benchmark solutions for diffusion and transport. In *Computational methods in transport: verification and validation, LNCSE-62*, pages 151–167. Springer, 2008.

[99] W. Zhang, L. Howell, A. Almgren, A. Burrows, J . Dolence, and J. Bell. Castro: A new compressible astrophysical solver. iii. multigroup radiation hydrodynamics. *Astrophysical Journal (supplement series)*, 204(7):27 pp, 2013.

[100] John I Castor. *Radiation hydrodynamics.* Cambridge University Press, 2004.

[101] Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *CoRR*, abs/2108.08481, 2021.

[102] Francesca Bartolucci, Emmanuel de BÃ©zenac, Bogdan RaoniÄ‡, Roberto Molinaro, Siddhartha Mishra, and Rima Alaifari. Are neural operators really neural operators? frame theory meets operator learning, 2023.

[103] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[104] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model reduction and neural networks for parametric pdes, 2021.

[105] Maarten V. de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M. Stuart. The cost-accuracy trade-off in operator learning with neural networks, 2022.

[106] Albert Clop, Daniel Faraco, and Alberto Ruiz. Stability of Calderón's inverse conductivity problem in the plane for discontinuous conductivities. *Inverse Problems & Imaging*, 4(1):49, 2010.

[107] Adrian Nachman. Reconstructions from boundary measurements. *Ann. Math.*, 128(3):531–576, 1988.

[108] Guillaume Bal and Alexandre Jollivet. Stability estimates in stationary inverse transport. *Inverse Problems & Imaging*, 2(4):427, 2008.

[109] William W Symes. The seismic reflection inverse problem. *Inverse problems*, 25(12):123008, 2009.

[110] Chengyuan Deng, Shihang Feng, Hanchen Wang, Xitong Zhang, Peng Jin, Yinan Feng, Qili Zeng, Yinpeng Chen, and Youzuo Lin. OpenFWI: Large-Scale Multi-Structural Benchmark Datasets for Seismic Full Waveform Inversion. *arXiv preprint arXiv:2111.02926*, 2021.

[111] Shitao Liu and Lauri Oksanen. A Lipschitz stable reconstruction formula for the inverse problem for the wave equation. *Transactions of the American Mathematical Society*, 368(1):319–335, 2016.

[112] Plamen Stefanov, Gunther Uhlmann, and Andras Vasy. On the stable recovery of a metric from the hyperbolic DN map with incomplete data. *Inverse Problems & Imaging*, 10(4):1141, 2016.

[113] Peter Caday, Maarten V de Hoop, Vitaly Katsnelson, and Gunther Uhlmann. Scattering control for the wave equation with unknown wave speed. *Archive for Rational Mechanics and Analysis*, 231(1):409–464, 2019.

[114] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Soc., 2010.

[115] J. Muller and S. Siltanen. *Linear and nonlinear inverse problems with practical applications.* SIAM, 2012.

[116] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[117] O. Boussif, D. Assouline, L. Benabbou, and Y. Bengio. MAgNet: Mesh Agnostic Neural PDE solver. *arXiv preprint arXiv:2210.05495*, 2022.

[118] J. Brandstetter, D. E. Worrall, and M. Welling. Message Passsing Neural PDE solvers. *arXiv preprint arXiv:2202.03376*, 2022.

[119] Samuel Lanthaler, Siddhartha Mishra, and George E Karniadakis. Error estimates for DeepONets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022.

[120] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for Fourier Neural Operators. *Journal of Machine Learning Research*, 22:Art–No, 2021.