Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs II: A class of inverse problems.

S. Mishra and R. Molinaro

Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs II: A class of inverse problems.

Siddhartha Mishra *and Roberto Molinaro †

June 29, 2020

Abstract

Physics informed neural networks (PINNs) have recently been very successfully applied for efficiently approximating inverse problems for PDEs. We focus on a particular class of inverse problems, the so-called data assimilation or unique continuation problems, and prove rigorous estimates on the generalization error of PINNs approximating them. An abstract framework is presented and conditional stability estimates for the underlying inverse problem are employed to derive the estimate on the PINN generalization error, providing rigorous justification for the use of PINNs in this context. The abstract framework is illustrated with examples of four prototypical linear PDEs. Numerical experiments, validating the proposed theory, are also presented.

1 Introduction

Partial Differential Equations (PDEs) are widely used in modeling of phenomena of interest in physics, biology and engineering. A wide variety of numerical methods, such as finite difference, finite element, finite volume and spectral methods, have been devised to numerically simulate solutions of different types of PDEs. Despite their widespread use, these methods may not suffice to solve many *large scale* problems with PDEs such as uncertainty quantification (UQ), PDEs in very high dimensions, deterministic and Bayesian inverse problems and PDEs with multiple scales and modeling multiphysics. A key challenge in the simulation of these problems is the very high computational cost, even on state of the art high performance computing platforms.

Deep learning, based on *deep neural networks* (DNNs) [22], i.e. functions formed by compositions of affine transformations and scalar non-linear activation functions, has been very successful in tackling diverse tasks in science and engineering [33] such as at image and text classification, computer vision, text and speech recognition, autonomous systems and robotics, game intelligence and even protein folding [18].

Deep learning is being increasingly used in the numerical approximations of PDEs. A brief and very incomplete survey of this rapidly growing body of literature is as follows, one approach in using deep neural networks 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 for approximation by DNNs. This approach is presented and analyzed for a variety of (parametric) elliptic, parabolic and linear transport PDEs in [5, 17, 23, 29] and references therein.

Another strategy is to augment existing numerical methods by adding deep learning based modules to them, for instance by learning free parameters of numerical schemes from data [44, 53] and references therein.

A third approach is to use deep neural networks to learn *observables* or quantities of interest of the solutions of the underlying PDEs, from data. This approach has been described in [37–39, 46] in the

---

*Seminar for Applied Mathematics (SAM), D-Math
ETH Zürich, Rämistrasse 101, Zürich-8092, Switzerland
†Seminar for Applied Mathematics (SAM), D-Math
ETH Zürich, Rämistrasse 101, Zürich-8092, Switzerland.

context of uncertainty quantification and PDE constrained optimization and [19] for model order reduction, among others.

As deep neural networks possess the so-called *universal approximation property* i.e any continuous function can be approximated by DNNs, see [3, 16, 24, 59] and references therein, it is natural to use them as ansatz spaces for the solutions of PDEs, in particular by collocating the PDE residual at suitably chosen training points. This approach was first proposed in [30, 31]. More recently, it has been revived and developed in significantly greater detail by the pioneering contributions of Karniadakis and collaborators, starting with [49, 50]. These authors have termed the underlying neural networks as *Physics Informed Neural Networks* (PINNs) and we will continue to use this by now widely accepted terminology in this paper. There has been an explosive growth of papers that present PINNs for different applications and a very incomplete list of references includes [15, 36, 42, 48, 51].

PINNs have already emerged as an attractive framework for approximating solutions of PDEs. They are particularly successful for approximating the so-called *inverse problems* involving PDEs, see [15, 42, 50, 51] 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. PINNs are very promising in solving inverse problems efficiently on account of their ability to integrate data and PDEs. This is seen in the robustness and accuracy of the results (see [42, 50, 51] and references therein). Another very attractive feature of PINNs is simplicity of their implementation. In particular, essentially the same code (with very minor modifications) can be used to approximate both forward and inverse problems.

Why are PINNs so efficient at approximating solutions to both forward and inverse problems for PDEs? Very few rigorous results on the approximation error with PINNs are available, with two recent notable exceptions. In [57], the authors investigate convergence of PINNs to solutions of the forward problem for linear elliptic and parabolic PDEs, on increasing the number of training samples. They prove convergence under certain assumptions on the training loss, based on Hölder regularized loss functions.

On the other hand, in another recent paper [45], the authors estimate the generalization (approximation) error for PINNs approximating forward problems for a large class of linear and nonlinear PDEs in terms of the training error and the number of training samples. The key point in [45] is the leveraging of *stability estimates* for classical solutions of the underlying PDE, together with estimates on quadrature errors, to derive upper bounds on the generalization error of PINNs.

The aforementioned articles provide rigorous justification for using PINNs to approximate forward problems for certain PDEs. However, to the best of the authors' knowledge, there is no rigorous estimate for the approximation error of PINNs for inverse problems. The main aim of this article 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 some inputs to the PDEs are unknown and need to be inferred from (possibly noisy) measurements of certain observables of the underlying solution. These are exactly the class of inverse problems, which were considered in recent papers [42, 50, 51] and references therein.

Our strategy for deriving bounds on the generalization errors follows the recent paper [45] and is based on *conditional stability estimates* on the underlying solution of the inverse problem. Such stability estimates have been widely investigated in the theoretical PDE literature and can be derived for a variety of linear and non-linear PDEs, for instance from the well-known *three-balls* inequalities [1] and references therein, and *Carleman estimates* [21] and references therein. We will formalize a very general framework for the unique continuation (data assimilation) inverse problem for an abstract PDE, with a wide variety of potential applications. Our derived generalization error estimate will show that the error due to approximating the underlying inverse problem with a trained PINN will be sufficiently low as long as

- The training error is low i.e, the PINN has been trained well. This error is computed and monitored during the training process. Hence, it is available *a posteriori*.

- The number of training (collocation) points is sufficiently large. This number is determined by the error due to an underlying quadrature rule.

- The solution to the underlying inverse problem is *conditionally stable* i.e, it is stable as long as the underlying solution and the approximating PINN are sufficiently regular.

Thus, with the derived error estimate, we identify possible mechanisms by which the PINNs are able to approximate inverse problems for PDEs so well and provide a rigorous justification for their successful use.

We will exemplify the proposed abstract framework by applying to four well-known model problems, namely the data assimilation problems for the Poisson, Heat, Wave and Stokes equations, covering the whole spectrum of linear elliptic, parabolic, hyperbolic and indefinite PDEs. For each of these examples, we derive bounds on the generalization error and also provide numerical experiments to validate the proposed theory.

The rest of the paper is organized as follows, in section 2, we describe the PINNs algorithm for the unique continuation (data assimilation) problem for an abstract PDE and derive an estimate on the generalization error. In sections 3,4,5 and 6, we consider the Poisson, Heat, Wave and Stokes equations, respectively.

2   An abstract framework for Physics informed Neural Networks approximating inverse problems for PDEs

2.1   The underlying abstract PDE

Let $\mathbb{D} \subset \mathbb{R}^{\bar{d}}$, for some $\bar{d} \geqslant 1$, be the underlying domain, with smooth (C^1) boundary, denoted by $\partial \mathbb{D}$. We include space-time domains by setting $\mathbb{D} = (0, T) \times D \subset \mathbb{R}^d$ with $d \geqslant 1$. In this case $\bar{d} = d + 1$ and the *space-time boundary* is $\partial \mathbb{D} = \partial D \times (0, T) \cup D \times \{t = 0\}$, with ∂D denoting the smooth boundary of D. Let $X = L^{p_x}(\mathbb{D}; \mathbb{R}^m)$, $Y = L^{p_y}(\mathbb{D}; \mathbb{R}^m)$ and $Z = L^{p_z}(\mathbb{D}'; \mathbb{R}^m)$, for some $\mathbb{D}' \subset \mathbb{D}$, with $m \geqslant 1$ and $1 \leqslant p_x, p_y, p_z < \infty$ be the underlying spaces and $X^* \subset X$, $Y^* \subset Y$, $Z^* \subset Z$ be Banach spaces. Let $\bar{X} \subset L^{\bar{p}_x}(\partial \mathbb{D}; \mathbb{R}^m)$ and $\bar{Y} \subset L^{\bar{p}_y}(\partial \mathbb{D}; \mathbb{R}^m)$, with $1 \leqslant \bar{p}_x, \bar{p}_y < \infty$ be Banach spaces.

2.1.1   The Forward problem

The abstract PDE is represented by the following differential equation,

$$\mathcal{D}(\mathbf{u}) = \mathbf{f}. \tag{2.1}$$

Here, the *differential operator* is a mapping, $\mathcal{D} : X^* \mapsto Y^*$ and the *input* $\mathbf{f} \in Y^*$, such that

$$
\begin{aligned}
(H1) : \quad & \|\mathcal{D}(\mathbf{u})\|_{Y^*} < +\infty, \quad \forall \, \mathbf{u} \in X^*, \; \text{with} \; \|\mathbf{u}\|_{X^*} < +\infty. \\
(H2) : \quad & \|\mathbf{f}\|_{Y^*} < +\infty.
\end{aligned}
\tag{2.2}
$$

The PDE (2.1) is equipped with the *boundary conditions*,

$$\mathcal{B}\left(\mathcal{T}\mathbf{u}\right) = \mathbf{f}_b, \tag{2.3}$$

with *trace operator* $\mathcal{T} : X^* \mapsto \bar{X}$ and a generic boundary operator $\mathcal{B} : \bar{X} \mapsto \bar{Y}$, with each of these operators being bounded under the corresponding norms. We note that the generic boundary condition also includes initial conditions when $\mathbb{D} = (0, T) \times D \subset \mathbb{R}^d$. We assume that $\mathbf{f}_b \in \bar{Y}$.

We further assume that the forward problem (2.1), (2.3) is well-posed i.e, given $\mathbf{f} \in Y^*, \mathbf{f}_b \in \bar{Y}$, there exists an unique $\mathbf{u} \in X^*$ such that (2.1) and (2.3) are satisfied.

2.1.2   The Inverse problem

As mentioned in the introduction, we will focus on the following inverse problem in this article, we assume that the generic boundary conditions (which include initial conditions) $\mathbf{f}_b$ in (2.3) are not known. Clearly, the forward problem for the PDE (2.1) will be ill-posed in this case. In particular, no unique solution can be guaranteed. Furthermore, we assume that we have access to (noiseless) measurements of the underlying solution $\mathbf{u}$ in a sub-domain $\mathbb{D}' \subset \mathbb{D}$ i.e,

$$\mathcal{L}(\mathbf{u}) = \mathbf{g}, \quad \forall x \in \mathbb{D}', \tag{2.4}$$

with the *restriction operator* $\mathcal{L} : X^* \mapsto Z^*$ and *data* $\mathbf{g} \in Z^*$, with

$$(H3): \quad \|\mathcal{L}(\mathbf{u})\|_{Z^*} < +\infty, \quad \forall\, \mathbf{u} \in X^*,\ \text{with } \|\mathbf{u}\|_{X^*} < +\infty.$$
$$(H4): \quad \|\mathbf{g}\|_{Z^*} < +\infty. \tag{2.5}$$

We term $\mathbb{D}'$ as the *observation domain*. Thus, in solving the inverse problem (2.1), (2.4), one determines the function $\mathbf{u} \in X^*$ and consequently the boundary conditions $\mathcal{T}(\mathbf{u})$ from the data (2.4), given only on the observation domain. This inverse problem is often referred to in the literature as the *unique continuation* or the *data assimilation* problem, [13] and references therein.

We further assume that solutions to the inverse problem (2.1), (2.4), satisfy the following *conditional stability estimate*,

Assumption 2.1. *Let $\hat{X} \subset X^* \subset X = L^{p_x}(\mathbb{D})$ be a Banach space. For any $\mathbf{u}, \mathbf{v} \in \hat{X}$, the differential operator $\mathcal{D}$ and restriction operator $\mathcal{L}$ satisfy,*

$$(H5): \quad \|\mathbf{u} - \mathbf{v}\|_{L^{p_x}(E)} \leqslant C_{pd} \left(\|\mathbf{u}\|_{\hat{X}}, \|\mathbf{v}\|_{\hat{X}} \right) \left(\|\mathcal{D}(\mathbf{u}) - \mathcal{D}(\mathbf{v})\|_Y^{\tau_p} + \|\mathcal{L}(\mathbf{u}) - \mathcal{L}(\mathbf{v})\|_Z^{\tau_d} \right), \tag{2.6}$$

for some $0 < \tau_p, \tau_d \leqslant 1$ and for any subset $\mathbb{D}' \subset E \subset \mathbb{D}$. This bound (2.6) is termed a *conditional stability estimate* as it presupposes that the underlying solutions have sufficiently regularity as $\hat{X} \subset X^* \subset X$.

Remark 2.2. We can extend the hypothesis for the inverse problem (2.1),(2.3) in the following ways,

- Allow the measurement set $\mathbb{D}'$ to intersect the boundary i.e, $\partial\mathbb{D}' \cap \partial\mathbb{D} \neq \Phi$.

- Replace the bound (2.6) with the weaker bound,

$$(H6): \quad \|\mathbf{u} - \mathbf{v}\|_{L^{p_x}(E)} \leqslant C_{pd} \left(\|\mathbf{u}\|_{\hat{X}}, \|\mathbf{v}\|_{\hat{X}} \right) \omega \left(\|\mathcal{D}(\mathbf{u}) - \mathcal{D}(\mathbf{v})\|_Y + \|\mathcal{L}(\mathbf{u}) - \mathcal{L}(\mathbf{v})\|_Z \right), \tag{2.7}$$

with $\omega : \mathbb{R} \mapsto \mathbb{R}_+$ being a modulus of continuity.

∎

We will provide several examples for this abstract Inverse problem satisfying the assumptions $H1 - H6$ in the following sections.

2.1.3 Literature on the unique continuation (data assimilation) problem

The unique continuation or data assimilation problem (2.1), (2.4), has been extensively studied by the theoretical PDE community for a long time, arguably starting with the influential work of Hadamard on the ill-posedness of the elliptic Cauchy problem. An extensive survey of the available literature on the unique continuation problem, particularly highlighting applications of the three balls inequality can be read from [1] and references therein. Similarly, the vast body of literature on use of Carleman estimates to study well-posedness of the unique continuation (data assimilation) problem for evolutionary PDEs has been surveyed in [21], with particularly emphasis on applications to controllability of PDEs.

Algorithms to numerically approximate solutions of the unique continuation problems are mostly based on quasi-reversibility methods [6, 32] and penalty methods [40], relying on Tikhonov regularization and its variants. More recently, regularized and stabilized non-confirming finite element methods have also been developed to deal with different examples of the data assimilation problem in [8, 9, 11–13] and references therein. We will provide an alternative approximation framework, based on physics informed neural networks (PINNs) in this paper.

2.2 Quadrature rules

In the following section, we will need to approximate integrals of functions. Hence, we need an abstract formulation for quadrature. To this end, we consider a mapping $h : \mathbb{D} \mapsto \mathbb{R}^m$, such that $h \in \underline{Y} \subset Y^* \subset Y$. We are interested in approximating the integral,

$$\overline{h} := \int_{\mathbb{D}} h(y)dy,$$

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

$$\overline{h}_N := \sum_{i=1}^{N} w_i h(y_i). \tag{2.8}$$

We further assume that the quadrature error is bounded as,

$$\left| \overline{h} - \overline{h}_N \right| \leqslant C_q \left(\|h\|_{\underline{Y}}, \bar{d} \right) N^{-\alpha}, \tag{2.9}$$

for some $\alpha > 0$.

As long as the domain $\mathbb{D}$ is in reasonably low dimension i.e $\bar{d} \leqslant 4$, we can use standard (composite) Gauss quadrature rules on an underlying grid. In this case, the quadrature points and weights depend on the underlying order of the quadrature rule [58] and the rate α depends on the regularity of the underlying integrand i.e, on the space $\underline{Y}$.

On the other hand, these grid based quadrature rules are not suitable for domains in high dimensions. For moderately high dimensions i.e $4 \leqslant \bar{d} \approx 20$, we can use *low discrepancy sequences*, such as the Sobol and Halton sequences, as quadrature points [14]. As long as the integrand h is of bounded *Hardy-Krause variation* [47], the error in (2.9) converges at a rate $(\log(N))^{\bar{d}} N^{-1}$. One can also employ sparse grids and Smolyak quadrature rules [7] in this regime.

For problems in very high dimensions $\bar{d} \gg 20$, Monte-Carlo quadrature is the numerical integration method of choice [14]. In this case, the quadrature points are randomly chosen, independent and identically distributed (with respect to a scaled Lebesgue measure). The estimate (2.9) holds in the root mean square (RMS) sense and the rate of convergence is $\alpha = \frac{1}{2}$.

We will also need integrals of functions $h_d : \mathbb{D}' \subset \mathbb{D}$ i.e,

$$\overline{h_d} := \int_{\mathbb{D}'} h_d(y) dy.$$

In principle, a different set of quadrature points $z_j \in \mathbb{D}'$ for $1 \leqslant j \leqslant N_d$, for some $N_d \in \mathbb{N}$ as well as weights w_j^d, can be chosen. Then the quadrature is defined by,

$$\overline{h}_{d,N} := \sum_{j=1}^{N_d} w_j^d h_d(z_j). \tag{2.10}$$

Analogous to (2.9), the resulting error is bounded by,

$$\left| \overline{h}_d - \overline{h}_{d,N} \right| \leqslant C_{qd} \left(\|h\|_{\underline{Z}}, \bar{d} \right) N_d^{-\alpha_d}, \tag{2.11}$$

for some $\alpha_d > 0$ and for $\underline{Z} \subset Z$.

2.3 PINNs

In this section, we will describe physics-informed neural networks (PINNs) for approximating solutions of the inverse problem (2.1), (2.4) in the following steps.

2.3.1 Neural Networks.

Given an input $y \in \mathbb{D}$, a feedforward neural network (also termed as a multi-layer perceptron), shown in figure 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 [22], resulting in the representation,

$$\mathbf{u}_\theta(y) = C_K \circ \sigma \circ C_{K-1} \ldots\ldots\ldots \circ \sigma \circ C_2 \circ \sigma \circ C_1(y). \tag{2.12}$$

Here, ∘ 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 [22]. Popular choices for the activation function σ in (2.12) include the sigmoid function, the hyperbolic tangent function and the *ReLU* function.

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

$$C_k z_k = W_k z_k + b_k, \quad \text{for } W_k \in \mathbb{R}^{d_{k+1} \times d_k}, z_k \in \mathbb{R}^{d_k}, b_k \in \mathbb{R}^{d_{k+1}}. \tag{2.13}$$

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

Thus in the terminology of machine learning (see also figure 1), our neural network (2.12) 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 (2.13) and then by a nonlinear (component wise) activation σ. A straightforward addition shows that our network contains $\left(\bar{d} + m + \sum_{k=2}^{K-1} d_k \right)$ neurons. We also denote,

$$\theta = \{W_k, b_k\}, \theta_W = \{W_k\} \quad \forall \, 1 \leqslant k \leqslant K, \tag{2.14}$$

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

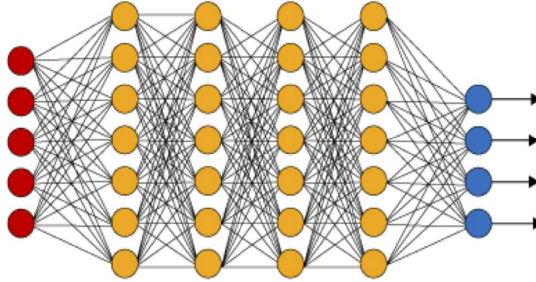$$M = \sum_{k=1}^{K-1} (d_k + 1) d_{k+1}. \tag{2.15}$$

Figure 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.

2.3.2 Training PINNS: Loss functions and optimization

The neural network $\mathbf{u}_\theta$ (2.12) depends on the tuning parameter $\theta \in \Theta$ of weights and biases. Within the standard paradigm of *deep learning* [22], one *trains* the network by finding tuning parameters θ such that the loss (error, mismatch, regret) between the neural network and the underlying target is minimized. Here, our target is the solution $\mathbf{u} \in X^*$ of the abstract inverse problem (2.1),(2.4) and we wish to find the tuning parameters θ such that the resulting neural network $\mathbf{u}_\theta$ approximates $\mathbf{u}$.

To do so, we follow [31, 45, 49] and define the following *PDE residual*:

$$\mathcal{R} = \mathcal{R}_\theta := \mathcal{D} (\mathbf{u}_\theta) - \mathbf{f}. \tag{2.16}$$

By assumptions (H1),(H2) (2.2), we see that $\mathcal{R} \in Y^*$ and $\|\mathcal{R}\|_{Y^*} < +\infty$ for all $\theta \in \Theta$. Note that $\mathcal{R}(\mathbf{u}) = \mathcal{D}(\mathbf{u}) - \mathbf{f} \equiv 0$, for the solution $\mathbf{u}$ of the PDE (2.1). Hence, the term *residual* is justified for (2.16).

As the inverse problem also involves a data term (2.4) on the observation domain, we need the following data residual:

$$\mathcal{R}_d = \mathcal{R}_{d, \theta} := \mathcal{L}(\mathbf{u}_\theta) - \mathbf{g}, \quad \forall x \in \mathbb{D}'. \tag{2.17}$$

By assumptions (H3),(H4) (2.5), we see that $\mathcal{R}_d \in Z^*$ and $\|\mathcal{R}_d\|_{Z^*} < +\infty$ for all $\theta \in \Theta$. Note that $\mathcal{R}_d(\mathbf{u}) = \mathcal{L}(\mathbf{u}) - \mathbf{g} \equiv 0$ for all $x \in \mathbb{D}'$, for the solution $\mathbf{u}$ of the inverse problem (2.1),(2.4).

The strategy of PINNs, following [50], is to minimize the *residuals* (2.16),(2.17), simultaneously over the admissible set of tuning parameters $\theta \in \Theta$ i.e

$$\text{Find } \theta^* \in \Theta : \quad \theta^* = \arg\min_{\theta \in \Theta} \left(\|\mathcal{R}_\theta\|_Y + \|\mathcal{R}_{d,\theta}\|_Z \right). \tag{2.18}$$

Realizing that $Y = L^{p_y}(\mathbb{D}), Z = L^{p_z}(\mathbb{D}')$ for some $1 \leqslant p_x, p_y < \infty$, we can equivalently minimize,

$$\text{Find } \theta^* \in \Theta : \quad \theta^* = \arg\min_{\theta \in \Theta} \left(\int_{\mathbb{D}} |\mathcal{R}_\theta(y)|^{p_y} dy + \int_{\mathbb{D}'} |\mathcal{R}_{d,\theta}(z)|^{p_z} dz \right). \tag{2.19}$$

As it will not be possible to evaluate the integral in (2.19) exactly, we need to approximate it numerically by a quadrature rule. To this end, we use the quadrature rules (2.8),(2.10) discussed earlier. We select the *training sets* $\mathcal{S}_{int} = \{y_i\}$ with $y_i \in \mathbb{D}$ for all $1 \leqslant i \leqslant N_{int}$ and $\mathcal{S}_d = \{z_j\}$ with $z_j \in \mathbb{D}'$ and $1 \leqslant j \leqslant N_d$ as the quadrature points for the quadrature rules (2.8), (2.10), respectively. We consider the following *loss function*:

$$J(\theta) := \sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta}(z_j)|^{p_z} + \lambda \sum_{i=1}^{N_{int}} w_i |\mathcal{R}_\theta(y_i)|^{p_y}, \tag{2.20}$$

with the residuals $\mathcal{R}_d$, $\mathcal{R}$ defined in (2.17) and (2.16). Here, $\lambda > 0$ is an additional hyperparameter that balances the roles of the data and PDE residuals.

It is common in machine learning [22] to regularize the minimization problem for the loss function i.e we seek to find,

$$\theta^* = \arg\min_{\theta \in \Theta} \left(J(\theta) + \lambda_{reg} J_{reg}(\theta) \right). \tag{2.21}$$

Here, $J_{reg} : \Theta \to \mathbb{R}$ is a *weight regularization* (penalization) term. A popular choice is to set $J_{reg}(\theta) = \|\theta_W\|_q^q$ for either $q = 1$ (to induce sparsity) or $q = 2$. The parameter $0 \leqslant \lambda_{reg} \ll 1$ balances the regularization term with the actual loss J (2.20).

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 solving this minimization problem approximately by either (first-order) stochastic gradient descent methods such as ADAM [27] or even higher-order optimization methods such as different variants of the LBFGS algorithm [20].

For notational simplicity, we denote the (approximate, local) minimum in (2.21) as θ^* and the underlying deep neural network $\mathbf{u}^* = \mathbf{u}_{\theta^*}$ will be our physics-informed neural network (PINN) approximation for the solution $\mathbf{u}$ of the PDE (2.1).

The proposed algorithm for computing this PINN is summarized below,

Algorithm 2.3. Finding a physics informed neural network to approximate the solution of the inverse problem (2.1),(2.4).

Inputs: *Underlying domains $\mathbb{D}' \subset \mathbb{D}$, differential operator $\mathcal{D}$ and input source term $\mathbf{f}$ for the PDE (2.1), restriction operator $\mathcal{L}$ and data term $\mathbf{g}$ for the data (2.4), quadrature points and weights for the quadrature rules (2.8) (2.10), non-convex gradient based optimization algorithms.*

Goal: *Find PINN $\mathbf{u}^* = \mathbf{u}_{\theta^*}$ for approximating the inverse problem for the (2.1) with data (2.4).*

Step 1: *Choose the training set $\mathcal{S}_{int} = \{y_i\}$ for $y_i \in \mathbb{D}$, for all $1 \leqslant i \leqslant N_{int}$ such that $\{y_i\}$ are quadrature points for the underlying quadrature rule (2.8). Choose the training set $\mathcal{S}_d = \{z_j\}$ for $z_j \in \mathbb{D}'$, for all $1 \leqslant j \leqslant N_d$ such that $\{z_j\}$ are quadrature points for the underlying quadrature rule (2.10).*

Step 2: *For an initial value of the weight vector $\overline{\theta} \in \Theta$, evaluate the neural network $\mathbf{u}_{\overline{\theta}}$ (2.12), the PDE residual (2.16), data residual (2.17), the loss function (2.21) and its gradients to initialize the underlying optimization algorithm.*

Step 3: *Run the optimization algorithm till an approximate local minimum θ^* of (2.21) is reached. The map $\mathbf{u}^* = \mathbf{u}_{\theta^*}$ is the desired PINN for approximating the solution $\mathbf{u}$ of the inverse problem (2.1),(2.4).*

2.4 An abstract estimate on the generalization error

In this section, we will estimate the error due to the PINN (generated by algorithm 2.3) in approximating the solution $\mathbf{u}$ of the inverse problem for PDE (2.1) with data (2.4).

We focus on the so-called *generalization error* of the PINN, which in machine learning terminology [56], is often understood as the error of the neural network on *unseen data*. Following the recent paper [45], we set $\mathbb{D}' \subset E \subset \mathbb{D}$ and define the corresponding generalization error as,

$$\mathcal{E}_G(E) = \mathcal{E}_G(E; \theta^*, \mathcal{S}_{int}, \mathcal{S}_d) := \|\mathbf{u} - \mathbf{u}^*\|_{L^{p_x}(E)}. \tag{2.22}$$

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 2.3. However, we shall suppress this explicit dependence for notational convenience.

As in the recent article [45], we estimate the generalization error (2.22) in terms of the so-called *training error* $\mathcal{E}_T = \mathcal{E}_T(\theta^*, \mathcal{S}_{int}, \mathcal{S}_d)$ defined by,

$$\mathcal{E}_T = \left(\mathcal{E}_{d,T}^{p_z} + \lambda \mathcal{E}_{p,T}^{p_y} \right)^{\frac{2}{p_y + p_z}}, \quad \mathcal{E}_{d,T} := \left(\sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta}(z_j)|^{p_z} \right)^{\frac{1}{p_z}}, \quad \mathcal{E}_{p,T} := \left(\sum_{i=1}^{N_{int}} w_i |\mathcal{R}_\theta(y_i)|^{p_y} \right)^{\frac{1}{p_y}}. \tag{2.23}$$

Note that, after the training has concluded, the training error $\mathcal{E}_T$ can be readily computed from the loss functions (2.20) or (2.21). The bound on generalization error in terms of training error is given by the following estimate,

Theorem 2.4. *Let $\mathbf{u} \in \hat{X} \subset X^* \subset X$ be the solution of the inverse problem associated with PDE (2.1) and data (2.4). Assume that the stability hypothesis (2.6) holds for any $\mathbb{D}' \subset E \subset \mathbb{D}$. Let $\mathbf{u}^* \in \hat{X} \subset X^*$ be a PINN generated by the algorithm 2.3, based on the training sets $\mathcal{S}_{int}$ (of quadrature points corresponding to the quadrature rule (2.8)) and $\mathcal{S}_d$ (of quadrature points corresponding to the quadrature rule (2.10)). Further assume that the residuals $\mathcal{R}_{\theta^*}$ (2.16), and $\mathcal{R}_{d,\theta^*}$ (2.17), be such that $|\mathcal{R}_{\theta^*}|^{p_y} \in \underline{Y}, |\mathcal{R}_{d,\theta^*}|^{p_z} \in \underline{Z}$ and the quadrature errors satisfy (2.9), (2.11). Then the following estimate on the generalization error (2.22) holds,*

$$\mathcal{E}_G \leqslant C_{pd} \left(\mathcal{E}_{p,T}^{\tau_p} + \mathcal{E}_{d,T}^{\tau_d} + C_q^{\frac{\tau_p}{p_y}} N_{int}^{-\frac{\alpha \tau_p}{p_y}} + C_{qd}^{\frac{\tau_d}{p_z}} N_d^{-\frac{\alpha_d \tau_d}{p_z}} \right), \tag{2.24}$$

with constants $C_{pd} = C_{pd} \left(\|\mathbf{u}\|_{\hat{X}}, \|\mathbf{u}^*\|_{\hat{X}} \right)$, $C_q = C_q \left(\||\mathcal{R}_{\theta^*}|^{p_y}\|_{\underline{Y}} \right)$, and $C_{qd} = C_{qd} \left(\||\mathcal{R}_{d,\theta^*}|^{p_z}\|_{\underline{Z}} \right)$ stem from (2.6), (2.9) and (2.11), respectively.

Proof. For notational simplicity, we denote $\mathcal{R} = \mathcal{R}_{\theta^*}$, the residual (2.16), corresponding to the trained neural network $\mathbf{u}^*$. Similarly, $\mathcal{R}_d = \mathcal{R}_{d,\theta^*}$ is the data residual (2.17), corresponding to $\mathbf{u}^*$.

As $\mathbf{u}$ solves the PDE (2.1) and $\mathcal{R}$ is defined by (2.16), we easily see that,

$$\mathcal{R} = \mathcal{D}(\mathbf{u}^*) - \mathcal{D}(\mathbf{u}). \tag{2.25}$$

Similarly, $\mathbf{u}$ satisfies the data relation (2.4) and by definition (2.17), we have,

$$\mathcal{R}_d = \mathcal{L}(\mathbf{u}^*) - \mathcal{L}(\mathbf{u}). \tag{2.26}$$

By our assumptions, PINN $\mathbf{u}^* \in \hat{X}$, hence, we can directly apply the *conditional stability estimate* (2.6) and use (2.25), (2.26) to obtain,

$$\begin{aligned}
\mathcal{E}_G(E) &= \|\mathbf{u} - \mathbf{u}^*\|_{L^{p_x}(E)} \quad \text{(by) (2.22)}, \\
&\leqslant C_{pd} \left(\|\mathcal{D}(\mathbf{u}^*) - \mathcal{D}(\mathbf{u})\|_Y^{\tau_p} + \|\mathcal{L}(\mathbf{u}) - \mathcal{L}(\mathbf{u}^*)\|_Z^{\tau_d} \right) \quad \text{(by) (2.6)}, \\
&\leqslant C_{pd} \left(\|\mathcal{R}\|_Y^{\tau_p} + \|\mathcal{R}_d\|_Z^{\tau_d} \right) \quad \text{(by) (2.25), (2.26)}.
\end{aligned} \tag{2.27}$$

By the fact that $Y = L^{p_y}(\mathbb{D})$, the definition of the training error (2.23) and the quadrature rule (2.8), we see that,

$$\|\mathcal{R}\|_Y^{p_y} \approx \sum_{i=1}^{N_{int}} w_i |\mathcal{R}_{\theta^*}|^{p_y} = \mathcal{E}_{p,T}^{p_y}.$$

Hence, the training error component $\mathcal{E}_{p,T}$ is a quadrature for the residual (2.16) and the resulting quadrature error, given by (2.9) translates to,

$$\|\mathcal{R}\|_Y^{p_y} \leqslant \mathcal{E}_{p,T}^{p_y} + C_q N_{int}^{-\alpha},$$

and as $\tau_p \leqslant 1$

$$\|\mathcal{R}\|_Y^{\tau_p} \leqslant \mathcal{E}_{p,T}^{\tau_p} + C_q^{\frac{\tau_p}{p_y}} N_{int}^{-\frac{\alpha\tau_p}{p_y}}. \tag{2.28}$$

Similarly as $Z = L^{p_z}(\mathbb{D}')$, the definition of the training error (2.23) and the quadrature rule (2.10), we have that,

$$\|\mathcal{R}_d\|_Z^{p_z} \approx \sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta^*}|^{p_z} = \mathcal{E}_{d,T}^{p_z}.$$

Hence, the training error component $\mathcal{E}_{d,T}$ is a quadrature for the residual (2.17) and the resulting quadrature error, given by (2.11) leads to,

$$\|\mathcal{R}_d\|_Z^{p_z} \leqslant \mathcal{E}_{d,T}^{p_z} + C_{qd} N_d^{-\alpha_d},$$

and as $\tau_d \leqslant 1$

$$\|\mathcal{R}_d\|_Z^{\tau_d} \leqslant \mathcal{E}_{d,T}^{\tau_d} + C_{qd}^{\frac{\tau_d}{p_z}} N_d^{-\frac{\alpha_d\tau_d}{p_z}}. \tag{2.29}$$

Substituting (2.28) and (2.29) into (2.27) yields the desired bound (2.24). □

We term a PINN, generated by the algorithm 2.3 to be *well-trained* if the following condition hold,

$$\max\left\{\mathcal{E}_{p,T}^{\tau_p}, \mathcal{E}_{d,T}^{\tau_d}\right\} \leqslant C_q^{\frac{\tau_p}{p_y}} N_{int}^{-\frac{\alpha\tau_p}{p_y}} + C_{qd}^{\frac{\tau_d}{p_z}} N_d^{-\frac{\alpha_d\tau_d}{p_z}}. \tag{2.30}$$

Thus, a well-trained PINN is one for which the training errors are smaller than the so-called *generalization gap* (given by the rhs of (2.30)).

The bound (2.24) leads directly to the following *consistency* lemma for a well-trained network,

Lemma 2.5. *Assume that for any N_{int}, N_d (size of the training sets $\mathcal{S}_{int}, \mathcal{S}_d$), the PINNs generated by the algorithm 2.3, denoted by $\mathbf{u}_{N_{int},N_d}^*$ are well-trained i.e, they satisfy (2.30). Furthermore, assume that there exists constant $\mathcal{C}$, independent of N_{int}, N_d such that the following holds,*

$$\max\left\{\left\|\mathbf{u}_{N_{int},N_d}^*\right\|_{\hat{X}}, \left\|\left|\mathcal{R}\left(\mathbf{u}_{N_{int},N_d}^*\right)\right|^{p_y}\right\|_{\underline{Y}}, \left\|\left|\mathcal{R}_d\left(\mathbf{u}_{N_{int},N_d}^*\right)\right|^{p_z}\right\|_{\underline{Z}}\right\} \leqslant \mathcal{C} < +\infty, \tag{2.31}$$

with $\mathcal{R}, \mathcal{R}_d$ being defined with respect to the trained PINN $\mathbf{u}_{N_{int},N_d}^*$ by (2.16), (2.17). Then we have,

$$\lim_{N_{int},N_d \to \infty} \mathbf{u}_{N_{int},N_d}^* = \mathbf{u}, \quad \text{in } L^{p_x}(E) \tag{2.32}$$

with $\mathbf{u}$ being the unique solution of the inverse problem, corresponding to the PDE (2.1) and data (2.4) and satisfying the conditional stability estimate (2.6), for any $\mathbb{D}' \subset E \subset \mathbb{D}$.

Thus, Lemma 2.5 provides consistency for the approximation of the solution of the inverse problem (2.1), (2.4) by PINNs. It can be thought of as a variant of the celebrated *Lax equivalence theorem* of classical numerical analysis as stability (in the form of the conditional stability estimate (2.6)) and accuracy (in the form of small training errors and decay of quadrature errors through (2.9) and (2.11)) imply convergence of the PINN to the solution of the inverse problem (2.1), (2.4).

The following remarks about Theorem 2.4 and Lemma 2.5 are in order.

Remark 2.6. The estimate (2.24) 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 (2.17), (2.16) can be approximated to high accuracy by the quadrature rules (2.8),(2.10). This regularity of PINNs can be enforced by choosing smooth activation functions such as the sigmoid and hyperbolic tangent in (2.12).

- Finally, the whole estimate (2.24) relies on the conditional stability estimate (2.6) for the inverse problem (2.1), (2.4). Thus, the generalization error estimate leverages conditional stability for inverse problems of PDEs into efficient approximation by PINNs. This is very similar to the program in a recent paper [45] for forward problems of PDEs.

$\blacksquare$

Remark 2.7. In addition to the requirement that the approximating PINNs are well-trained (2.30), Lemma 2.5 requires the uniform bound (2.31) on the PINNs. Given the structure of the PINN (2.12), in principle this bound can be explicitly estimated in terms of the weights in (2.12) by successive differentiation. Consequently, one can penalize this bound on weights, in a manner analogous to the regularization term in (2.21) and ensure (2.31). However, given the complicated algebraic expressions that result in this process, we will not explore this further but rather verify these assumptions a posteriori. $\blacksquare$

Remark 2.8. The estimate (2.24) was based on deterministic quadrature rules (2.8), (2.10). Following section 2.4.1 of recent paper [45], it can be extended, in a straightforward manner, to the case of randomly chosen training points that stem from Monte Carlo quadrature. $\blacksquare$

Remark 2.9. We have considered a noiseless measurement in the data term (2.4) on the observation domain. However, the bound (2.24) 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,

$$\mathcal{L}(\mathbf{u}) = \mathbf{g} + \eta, \tag{2.33}$$

for a noise term $\eta \in Z$. Then, a straightforward modification of the argument in the proof of Theorem 2.4 yields the bound,

$$\mathcal{E}_G \leqslant C_{pd} \left(\mathcal{E}_{p,T}^{\tau_p} + \mathcal{E}_{d,T}^{\tau_d} + \|\eta\|_{L^{p_z}(\mathbb{D}')}^{\tau_d} + C_q^{\frac{\tau_p}{p_y}} N_{int}^{-\frac{\alpha\tau_p}{p_y}} + C_{qd}^{\frac{\tau_d}{p_z}} N_d^{-\frac{\alpha_d\tau_d}{p_z}} \right), \tag{2.34}$$

with constants already defined in (2.24). 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_{int}, N_d sufficiently small, the noise term will dominate (2.34) and one has to use a Bayesian framework to approximate solutions of the inverse problem in this regime. $\blacksquare$

3 Poisson's equation

As a first example for the abstract inverse problem (2.1), (2.4), we consider the Poisson's equation as a model problem for linear elliptic PDEs.

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,

$$-\Delta u = f, \quad \forall x \in D, \tag{3.1}$$

---

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

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 (3.1) in the following weak sense,

$$\int_D \nabla u \cdot \nabla v \, dx = \int_D f v \, dx, \tag{3.2}$$

for all test functions $v \in H_0^1(D)$. Note that (3.2) follows as a consequence of multiplying the test function $v \in H_0^1(D)$ and integrating by parts. Moreover, the PDE (3.1) 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,

$$u|_{D'} = g, \quad \text{for some } D' \subset D, \tag{3.3}$$

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

Formally, for the unique continuation problem (3.1), (3.3), to have a solution, it is necessary that g satisfies the Poisson's equation (3.1) in D'. Hence, the unique continuation problem is formally equivalent to,

$$\begin{aligned}
-\Delta u &= f, \quad \forall x \in D \setminus D', \\
u &= g, \quad \forall x \in \partial D', \\
\nabla u \cdot \mathbf{n} &= \nabla g \cdot \mathbf{n}, \quad \forall x \in \partial D'.
\end{aligned} \tag{3.4}$$

The problem (3.4) 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 (3.1), (3.3) are classical, see [1] for a detailed survey. Here, we follow the slightly simplified presentation due to [13] and state the following result,

Theorem 3.1. *[[13], Theorem 7.1]: Let $f \in L^2(D)$ and let $u \in H^1(D)$ such that (3.2) holds for all test functions $v \in H_0^1(D)$. Let $g \in H^1(D')$ such that (3.3) holds, then for every open simply connected set $E \subset D$ such that $\mathrm{dist}(E, \partial D) > 0$, there holds,*

$$\|u\|_{H^1(E)} \leqslant C \left(\|u\|_{H^1(D)} \right) \omega \left(\|f\|_{L^2(D)} + \|g\|_{L^2(D')} \right). \tag{3.5}$$

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,

$$\|u\|_{H^1(D)} \leqslant C \left(\|u\|_{H^1(D)} \right) \omega \left(\|f\|_{L^2(D)} + \|g\|_{L^2(D')} \right), \tag{3.6}$$

with the same $C(R)$ as in (3.5), but with modulus of continuity given by $\omega(R) = |\log(R)|^{-\tau}$, with $\tau \in (0, 1)$.

The theorem, as presented in [13] (Theorem 7.1) has slightly better estimates than (3.5), (3.6). In particular, on the rhs, 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 [1] and the proof of (3.5), (3.6) is based on the *three-balls* inequality.

Remark 3.2. Relating the above formulation of the unique continuation inverse problem (3.1), (3.3) to the abstract formalism presented in section 2, we see that with $X = L^2(D), Y = L^2(D), Z = L^2(D')$, the differential operator $\mathcal{D} = -\Delta$, interpreted weakly, and the observable $\mathcal{L} = ID$, it is straightforward to bound the abstract conditional stability estimate (2.6) in the concrete forms (3.5) or (3.6). Thus, this problem falls squarely in the framework considered in section 2 and can be approximated by PINNs. ∎

3.2 PINNs

In order to complete the algorithm 2.3 to generate a PINN for approximating the inverse problem (3.1), (3.3), we need the following ingredients,

3.2.1 Training sets

As the training set S_{int} in algorithm 2.3, we take a set of quadrature points $y_i \in D$, for $1 \leqslant i \leqslant N_{int}$, corresponding to the quadrature rule (2.8). 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 $S_d = \{z_j\}$ for $z_j \in D'$, with $1 \leqslant j \leqslant N_d$, are quadrature points, corresponding to the quadrature rule (2.10).

3.2.2 Residuals

We will require that for parameters $\theta \in \Theta$, the neural networks $u_\theta \in C^k(D)$, for $k \geqslant 2$. This can be enforced by choosing a sufficiently regular activation function in (2.12). We define the following residuals that are needed in algorithm 2.3. The PDE residual (2.16) is given by,

$$\mathcal{R}_\theta = -\Delta u_\theta - f, \quad \forall x \in D, \tag{3.7}$$

and the data residual (2.17) on the observation domain is given by,

$$\mathcal{R}_{d,\theta} = u_\theta - g, \quad \forall x \in D'. \tag{3.8}$$

3.2.3 Loss functions

In algorithm 2.3 for approximating the inverse problem (3.1), (3.3), we will need the following loss function,

$$J(\theta) = \sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta}(z_j)|^2 + \lambda \sum_{i=1}^{N_{int}} w_i |\mathcal{R}_\theta(y_i)|^2, \tag{3.9}$$

with hyperparamter λ, residuals defined in (3.7), (3.8), training points defined above and weights w_i, w_j^d, corresponding to quadrature rules (2.8) and (2.10), respectively.

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 2.3 with training sets, residuals and loss functions described above, as

$$\mathcal{E}_G(E) = \|u - u^*\|_{H^1(E)}. \tag{3.10}$$

As in theorem 2.4, this generalization error will be estimated in terms of the following training errors,

$$\mathcal{E}_{d,T} = \left(\sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta^*}(z_j)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_{p,T} = \left(\sum_{i=1}^{N_{int}} w_i |\mathcal{R}_{\theta^*}(y_i)|^2 \right)^{\frac{1}{2}}. \tag{3.11}$$

Note that the training errors $\mathcal{E}_{p,T}$ and $\mathcal{E}_{d,T}$, can be readily computed from the loss functions (2.21), (3.9), a posteriori. We have the following estimate on the generalization error in terms of the training error,

Lemma 3.3. *For $f \in C^{k-2}(D)$ and $g \in C^k(D')$, with continuous extensions of the functions and derivatives upto the boundaries of the underlying sets and with $k \geqslant 2$, Let $u \in H^1(D)$ be the solution of the inverse problem corresponding to the Poisson's equation (3.1) i.e, it satisfies (3.2) for all test functions $v \in H_0^1(D)$ and satisfies the data (3.3). Let $u^* = u_{\theta^*} \in C^k(D)$ be a PINN generated by the algorithm 2.3, with loss functions (2.21), (3.9). Then, the generalization error (3.10) for any any $E \subset D$, with E open, simply connected and such that $dist(E, \partial D) > 0$ is bounded by,*

$$\|u - u^*\|_{H^1(E)} \leqslant C \left(\|u\|_{H^1(D)}^{1-\tau} + \|u^*\|_{H^1(D)}^{1-\tau} \right) \left(\mathcal{E}_{p,T}^\tau + \mathcal{E}_{d,T}^\tau + C_q^{\frac{\tau}{2}} N_{int}^{-\frac{\alpha\tau}{2}} + C_{qd}^{\frac{\tau}{2}} N_d^{-\frac{\alpha_d\tau}{2}} \right), \tag{3.12}$$

for some $\tau \in (0, 1)$ and constant C depending on E and with constants $C_q = C_q\left(\|\mathcal{R}\|_{C^{k-2}(D)}\right)$ and

$C_{qd} = C_{qd}\left(\|\mathcal{R}\|_{C^k(D')}\right)$, *given by the quadrature error bounds (2.9), (2.11), respectively.*

Moreover, we also have the global error bound,

$$\|u - u^*\|_{H^1(D)} \leqslant C\left(\|u\|_{H^1(D)}^{1-\tau} + \|u^*\|_{H^1(D)}^{1-\tau}\right)\left|\log\left(\mathcal{E}_{p,T} + \mathcal{E}_{d,T} + C_q^{\frac{1}{2}}N_{int}^{-\frac{\alpha}{2}} + C_{qd}^{\frac{1}{2}}N_d^{-\frac{\alpha_d}{2}}\right)\right|^{-\tau}. \tag{3.13}$$

Proof. For notational simplicity, we denote $\mathcal{R} = \mathcal{R}_{\theta^*}$ and $\mathcal{R}_d = \mathcal{R}_{d,\theta^*}$.

Define $\hat{u} = u^* - u \in H^1(D)$, by linearity of the differential operator in (3.1) and the data observable in (3.3) and by definitions (3.7), (3.8), we see that $\hat{u}$ satisfies,

$$\begin{aligned} -\Delta\hat{u} &= \mathcal{R}, \quad \forall x \in D, \\ \hat{u} &= \mathcal{R}_d, \quad \forall x \in D', \end{aligned} \tag{3.14}$$

with Poisson equation being satisfied in the following weak sense,

$$\int_D \nabla\hat{u} \cdot \nabla v \, dx = \int_D \mathcal{R}v \, dx, \tag{3.15}$$

for all test functions $v \in H_0^1(D)$. Hence, we can directly apply the conditional stability estimate (3.5) to obtain for some $\tau \in (0, 1)$,

$$\begin{aligned} \|\hat{u}\|_{H^1(E)} &\leqslant C(\|\hat{u}\|_{H^1(D)}^{1-\tau})\left(\|\mathcal{R}_d\|_{L^2(D')} + \|\mathcal{R}\|_{L^2(D)}\right)^{\tau} \\ &\leqslant C\left(\|u\|_{H^1(D)}^{1-\tau} + \|u^*\|_{H^1(D)}^{1-\tau}\right)\left(\|\mathcal{R}_d\|_{L^2(D')} + \|\mathcal{R}\|_{L^2(D)}\right)^{\tau}. \end{aligned} \tag{3.16}$$

Recognizing that the training errors $\mathcal{E}_{d,T}^2$ and $\mathcal{E}_{p,T}^2$ are the quadratures for $\|\mathcal{R}_d\|_{L^2(D')}^2$ and $\|\mathcal{R}\|_{L^2(D)}^2$ with respect to the quadrature rules (2.10) and (2.8), respectively and using bounds (2.11) and (2.9) yields,

$$\begin{aligned} \|\mathcal{R}_d\|_{L^2(D')} &\leqslant \mathcal{E}_{d,T} + C_{qd}^{\frac{1}{2}}N_d^{\frac{\alpha_d}{2}}, \\ \|\mathcal{R}\|_{L^2(D)} &\leqslant \mathcal{E}_{p,T} + C_q^{\frac{1}{2}}N_{int}^{\frac{\alpha}{2}}, \end{aligned} \tag{3.17}$$

with constants $C_q = C_q\left(\|\mathcal{R}\|_{C^{k-2}(D)}\right)$ and $C_{qd} = C_{qd}\left(\|\mathcal{R}_d\|_{C^k(D')}\right)$.

It is straightforward to obtain the desired inequality (3.12) by substituting (3.17) into (3.16).

The bound (3.13) can be obtained, completely analogously, by replacing (3.5) in (3.16) with (3.6). $\square$

$K-1$	$\tilde{d}$	q	λ_{reg}	λ
4, 8, 10	16, 20, 24	2	0, 10^{-6}	0.001, 0.01, 0.1, 1

Table 1: Hyperparamters configurations used for ensemble training inn all the numerical experiments.

3.4 Numerical experiments

We follow [13] and repeat their experiment from section 7.4.3. As in [13], the unique continuation problem for the Poisson equation is solved by considering the PDE (3.1) in the domain $D = (0, 1)^2$ (see figure 2), with source term,

$$f(x_1, x_2) = -60\left(x_1 - x_1^2 + x_2 - x_2^2\right). \tag{3.18}$$

The data (3.3) will be given in the observation domain (see figure 2),

$$D' = \{(x_1, x_2) \in \mathbb{R}^2 : |x_1 - 0.5| < 0.375; \ |x_2 - 0.5| < 0.375\}. \tag{3.19}$$
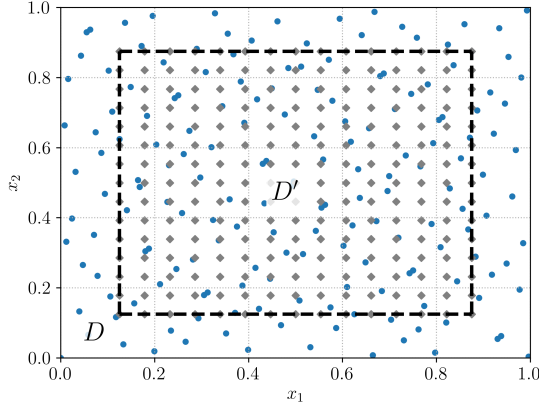
Figure 2: The domains D, D' for the Poission's equation numerical experiment. Training set $\mathcal{S}_{int}$ are Sobol points (blue dots) and training set $\mathcal{S}_d$ are Cartesian grid points (grey squares).

In order to generate the data term g in (3.3), we find that,

$$u(x_1, x_2) = 30x_1x_2(1 - x_1)(1 - x_2), \tag{3.20}$$

is an exact solution of the Poisson equation (3.1), with the source term (3.18) and let $g = u|_{D'}$, with D' (3.19), as the data term in (3.3). Clearly the exact solution u (3.20) and inputs f, g are as regular as required in Lemma 3.3.

(a) Exact Solution u

(b) PINN u^*
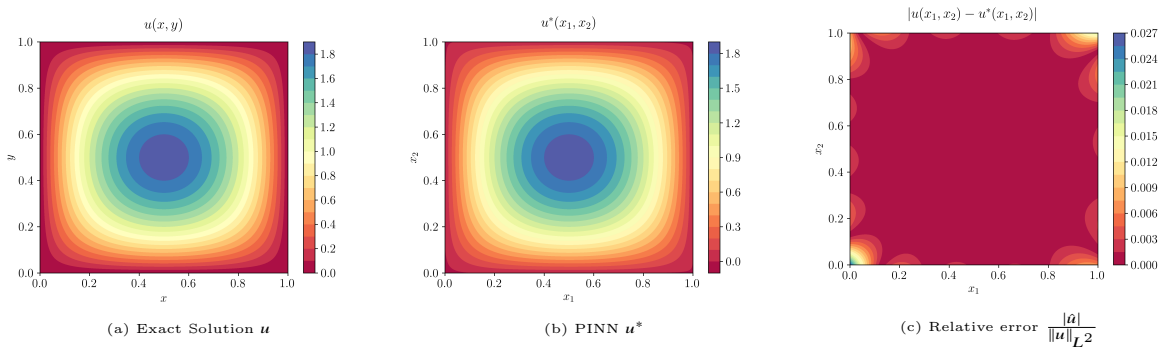
(c) Relative error $\frac{|\hat{u}|}{\|u\|_{L^2}}$

Figure 3: 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 2.3, with loss functions (3.9), corresponding to training sets $\mathcal{S}_{int}, \mathcal{S}_d$. For $\mathcal{S}_{int}$, we will use Sobol points on the domain D and for $\mathcal{S}_d$, we set up a Cartesian grid on the inner square D' (3.19), see figure 2 for a representation of both training sets. Note that a simple midpoint rule is used as the quadrature rule (2.10), whereas a quasi-Monte Carlo integration is used in (2.8).

As the PINNs in algorithm 2.3, contain several hyperparameters namely, the number of hidden layers $K - 1$ (depth) in (2.12), number of neurons $d_k \equiv \tilde{d}$ (width) in each hidden layer, the exponent q of the regularization term in loss function (2.21), the size λ_{reg} of the regularization term and the hyperparameter λ in the loss function (3.9). We follow [39] and perform an *ensemble training*, with the hyperparameter range specified in Table 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 value of the average training loss $\mathcal{E}_T = \sqrt{\mathcal{E}_{d,T}^2 + \lambda \mathcal{E}_{p,T}^2}$ over the retrainings.

We set $r = \text{meas}(D') = 9/16$ and for any $N = N_{int} + N_d$, we set $N_d = rN$ and present results for the PINNs in figure 3, where we plot the exact solution u (3.20), the PINN solution u^* and the error $\hat{u} = u^* - u$,

N	$K-1$	$\tilde{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\|u-u^*\|_{L^2}$	$\|u-u^*\|_{H^1}$
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 2: Poisson's equation: relative percentage generalization errors and training errors for different numbers of training points.

N	$K-1$	$\tilde{d}$	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 3: Poisson's equation with noisy measurements: relative percentage generalization errors and training errors for different numbers of training points.

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 (3.9) and $\lambda_{reg} = 0$ in (2.21). We observe from this figure, that already for this very low number of training points, the PINN generated by algorithm 2.3 is able to approximate the underlying solution of the inverse problem (3.1), (3.3), very accurately. The plot of the error in Figure 3 (right) shows that the error is mostly concentrated near the boundary and shows a logarithmic behavior, as predicted by the theory and seen in [13].

The efficiency of PINNs in approximating solutions to this problem is reinforced from Table 2, where we present the (percentage relative) generalization errors $\mathcal{E}_G(D)$ (3.10) 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 (3.13). 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 2.9, we can also extend the bounds (3.12), (3.13), to the case of noisy data, analogous to (2.34). To test the validity of PINNs in this regime, we perturb the right hand side of the data term (3.3), with 1% standard normal noise, run algorithm 2.3 to generate PINNs for approximating the inverse problem for the Poisson equation and present the results in Table 3. From this table, we see that although slightly higher than in the noise-free case (compare with Table 2), 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.

4 Heat Equation

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

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,

$$u_t - \Delta u = f, \quad \forall x \in D, t \in (0, T), \tag{4.1}$$

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 (4.1) in a weak sense.

The heat equation (4.1) 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*:

$$u = g, \quad \forall (x, t) \in D' \times (0, T), \tag{4.2}$$

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 (4.1), (4.2), amounts to finding the solution u of the heat equation (4.1) 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 [25], based on the well-known Carleman estimates,

Theorem 4.1. *[25]: Let $u \in H^1((0, T); H^{-1}(D)) \cap L^2((0, T); H^1(D))$ solve the heat equation (4.1) with source term $f \in L^2(D_T)$, then for every $0 \leqslant \bar{T} < T$, there holds,*

$$\|u\|_{C([\bar{T}, T]; L^2(D))} + \|u\|_{L^2((0,T); H^1(D))} \leqslant C \left(\|u\|_{L^2(D'_T)} + \|f\|_{L^2(D_T)} + \|u\|_{L^2(\partial D \times (0,T))} \right), \tag{4.3}$$

for some constant $C > 0$.

Remark 4.2. Relating the above formulation of the data assimilation inverse problem (4.1), (4.2) to the abstract formalism presented in section 2, we readily see that $X = Y = L^2(D_T), Z = L^2(D'_T)$, the differential operator is $\mathcal{D} = \partial_t - \Delta$, interpreted weakly, and the observable is $\mathcal{L} = ID$. Moreover, the abstract conditional stability estimate (2.6) takes the concrete form (4.3). Thus, this problem falls squarely in the framework considered in section 2 and can be approximated by PINNs. ∎

4.2 PINNs

We specify the algorithm 2.3 to generate a PINN for approximating the data assimilation inverse problem (4.1), (4.2), in the following steps,

4.2.1 Training sets

The training set $\mathcal{S}_d = \{z_j\}$ for $z_j = (x, t)_j \in D'_T$, with $1 \leqslant j \leqslant N_d$, are quadrature points, corresponding to the quadrature rule (2.10). Given the fact that we also consider boundary conditions in (4.1), we need to slightly modify the training set on which the PDE residual (2.16) is going to be collocated. We take *interior* training set $\mathcal{S}_{int}$ as set of quadrature points $y_i = (x, t)_i \in D_T$, for $1 \leqslant i \leqslant N_{int}$, corresponding to the quadrature rule (2.8). 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 $\mathcal{S}_{sb} = \{\bar{y}_i\}$ for $1 \leqslant i \leqslant N_{sb}$, with $\bar{y}_i = (\bar{x}, t)_i$, 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 $h : \partial D \times (0, T) \mapsto \mathbb{R}$, we approximate

$$\int_0^T \int_{\partial D} h(x, t) d\sigma(x) dt \approx \sum_{i=1}^{N_{sb}} w_i^{sb} h(\bar{x}_i, t_i), \tag{4.4}$$

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

$$\left| \int_0^T \int_{\partial D} h(x,t) d\sigma(x) dt - \sum_{i=1}^{N_{sb}} w_i^{sb} h(\bar{x}_i, t_i) \right| \leqslant C_{bd} N_{sb}^{-\alpha_{sb}}, \tag{4.5}$$

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

4.2.2 Residuals

We will require that for paramters $\theta \in \Theta$, the neural networks $(x,t) \mapsto u_\theta(x,t) \in C^k(\overline{D_T})$, for $k \geqslant 2$. We define the following residuals that are needed in algorithm 2.3. The PDE residual (2.16) is given by,

$$\mathcal{R}_\theta = \partial_t u_\theta - \Delta u_\theta - f, \quad \forall (x,t) \in D_T. \tag{4.6}$$

We need the following residual to account for the boundary data in (4.1),

$$\mathcal{R}_{sb,\theta} = u_\theta|_{\partial D \times (0,T)}, \tag{4.7}$$

and the data residual (2.17) is given by,

$$\mathcal{R}_{d,\theta} = u_\theta - g, \quad \forall (x,t) \in D_T'. \tag{4.8}$$

4.2.3 Loss functions

In algorithm 2.3 for approximating the inverse problem (4.1), (4.2), we will need the following loss function,

$$J(\theta) = \sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta}(z_j)|^2 + \sum_{i=1}^{N_{sb}} w_i^{sb} |\mathcal{R}_{sb,\theta}(\bar{y}_i)|^2 + \lambda \sum_{i=1}^{N_{int}} w_i |\mathcal{R}_\theta(y_i)|^2, \tag{4.9}$$

with hyperparamter λ, residuals defined in (4.6), (4.7), (4.8), training points defined above and weights w_i, w_i^{sb}, w_j^d, corresponding to quadrature rules (2.8), (4.4),and (2.10), respectively.

4.3 Estimates on the generalization error

For any $0 \leqslant \bar{T} < T$, we define the generalization error for the PINN $u^* = u_{\theta^*}$, generated by algorithm 2.3 to approximate the solution u to the data assimilation, inverse problem (4.1), (4.2) as,

$$\mathcal{E}_G(\bar{T}) = \|u - u^*\|_{C([\bar{T},T];L^2(D))} + \|u - u^*\|_{L^2((0,T);H^1(D))}. \tag{4.10}$$

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

$$\mathcal{E}_{d,T} = \left(\sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta^*}(z_j)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_{int,T} = \left(\sum_{i=1}^{N_{int}} w_i |\mathcal{R}_{\theta^*}(y_i)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_{sb,T} = \left(\sum_{i=1}^{N_{sb}} w_i^{sb} |\mathcal{R}_{\theta^*}(\bar{y}_i)|^2 \right)^{\frac{1}{2}}. \tag{4.11}$$

Note that the training errors $\mathcal{E}_{int,T}, \mathcal{E}_{sb,T}$ and $\mathcal{E}_{d,T}$, can be readily computed from the loss functions (2.21), (4.9), a posteriori. We have the following estimate on the generalization error in terms of the training error,

Lemma 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 $k \geqslant 2$, 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 (4.1) and satisfies the data (4.2). Let $u^* = u_{\theta^*} \in C^k(D_T)$ be a PINN generated by the algorithm 2.3, with loss functions (2.21), (4.9). Then, the generalization error (4.10) for any $0 \leqslant \bar{T} < T$ is bounded by,*

$$\mathcal{E}_G(\bar{T}) \leqslant C \left(\mathcal{E}_{d,T} + \mathcal{E}_{int,T} + \mathcal{E}_{sb,T} + C_q^{\frac{1}{2}} N_{int}^{-\frac{\alpha}{2}} + C_{bd}^{\frac{1}{2}} N_{sb}^{-\frac{\alpha_{sb}}{2}} + C_{qd}^{\frac{1}{2}} N_d^{-\frac{\alpha_d}{2}} \right), \tag{4.12}$$

for some constant C depending on $\bar{T}, u, u^*$ and with constants $C_q = C_q\left(\|\mathcal{R}_{\theta^*}\|_{C^{k-2}(D_T)} \right)$, $C_{bd} = C_{bd}\left(\|\mathcal{R}_{sb,\theta^*}\|_{C^k(\partial D \times (0,T))} \right)$ and $C_{qd} = C_{qd}\left(\|\mathcal{R}_{d,\theta^*}\|_{C^k(D_T')} \right)$, given by the quadrature error bounds (2.9), (4.5) and (2.11), respectively.

Proof. For notational simplicity, we denote $\mathcal{R} = \mathcal{R}_{\theta^*}, \mathcal{R}_{sb} = \mathcal{R}_{sb,\theta^*}$ and $\mathcal{R}_d = \mathcal{R}_{d,\theta^*}$.

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 (4.1) and the data observable in (4.2) and by definitions (4.6), (4.7) and (4.8), we see that $\hat{u}$ satisfies,

$$\begin{aligned}
\hat{u}_t - \Delta\hat{u} &= \mathcal{R}, \quad \forall x,t \in D_T, \\
\hat{u} &= \mathcal{R}_{sb}, \quad \forall x \in \partial D, t \in (0,T), \\
\hat{u} &= \mathcal{R}_d, \quad \forall x,t \in D_T'.
\end{aligned} \tag{4.13}$$

Hence, we can directly apply the conditional stability estimate (4.3) to obtain,

$$\|\hat{u}\|_{C([\bar{T},T];L^2(D))} \leqslant C \left(\|\mathcal{R}_d\|_{L^2(D')_T} + \|\mathcal{R}\|_{L^2(D_T)} + \|\mathcal{R}_{sb}\|_{L^2(\partial D \times (0,T))} \right), \tag{4.14}$$

with the constant C from (4.3).

Recognizing that the training errors $\mathcal{E}_{d,T}^2, \mathcal{E}_{sb,T}^2$ and $\mathcal{E}_{int,T}^2$ are the quadrature approximations for $\|\mathcal{R}_d\|_{L^2(D')_T}^2$, $\|\mathcal{R}_{sb}\|_{L^2(\partial D \times (0,T))}^2$ and $\|\mathcal{R}\|_{L^2(D_T)}^2$ with respect to the quadrature rules (2.10), (4.4) and (2.8), respectively and using bounds (2.11),(4.5) and (2.9) yields the inequality (3.17). Substituting it into (4.14) leads to the desired bound (4.12) in a straightforward manner. □
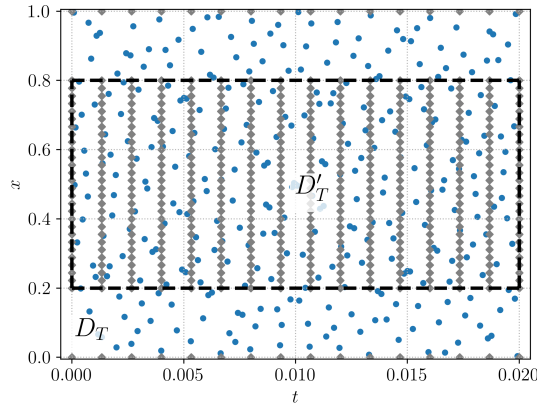

Figure 4: The domains D_T, D_T' for the heat equation numerical experiment. Training set $\mathcal{S}_{int}$ are Sobol points (blue dots) and training set $\mathcal{S}_d \cup \mathcal{S}_{sb}$ are Cartesian grid points (grey squares).

(a) Exact Solution u

(b) PINN u^*

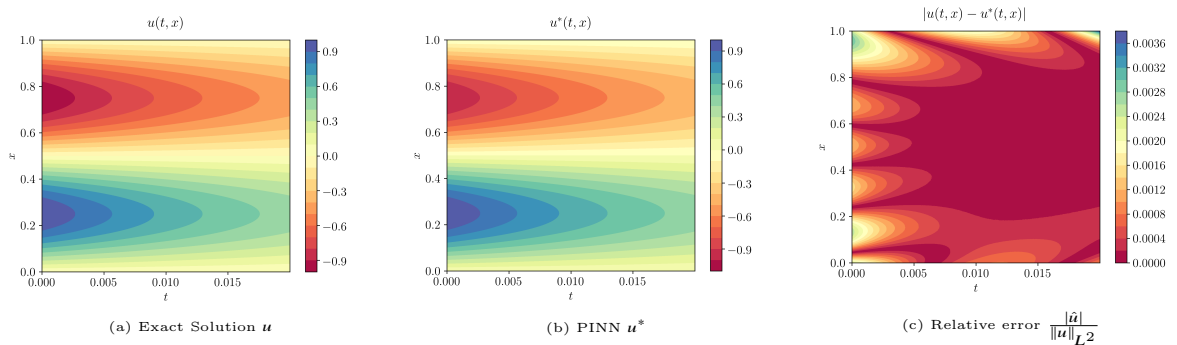(c) Relative error $\frac{|\hat{u}|}{\|u\|_{L^2}}$

Figure 5: 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.

18

4.4 Numerical experiments

4.4.1 Heat equation in one space dimension

We follow [12] and start by considering the heat equation (4.1) 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 (4.2) is specified in the observation domain D_T', with $D' = (a, 1-a)$, and as in [12], we set $a = 0.2$. The data term g and the source term f are generated from the exact solution,

$$u(x,t) = e^{-4\pi^2 t^2} \sin(2\pi x). \tag{4.15}$$

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 4 for an illustration). We run the algorithm 2.3 on these training sets to generate the PINN for approximating the data assimilation problem for the heat equation (4.1), (4.2).

In figure 5, 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 4. 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 (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 (4.12) and can be readily bounded above by the lhs of (4.12). 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 (4.9).

N	$K-1$	$\tilde{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\|u-u^*\|_{L^2}$	$\|u-u^*\|_{H^1}$
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 4: 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 (4.15)), 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 2.3. Note that we can readily prove a version of the generalization error estimate (4.12) in this setting by adapting the arguments in [45] (see Lemma 2.10) and also the recent paper [38]. The resulting (averaged over $K = 30$ different randomly chosen training sets) generalization errors in L^2 and H^1 are shown in Table 5. 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.

4.4.2 Heat equation in several space dimensions

For this experiment, we consider the linear heat equation (4.1) in the domain $D_T = [0,1] \times [0,1]^n$ for different space dimensions n. We follow the example in [45] and consider the heat equation with initial

N	$K-1$	$\tilde{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\lvert\lvert u-u^*\rvert\rvert_{L^2}$	$\lvert\lvert u-u^*\rvert\rvert_{H^1}$
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: 1-D Heat equation with randomly chosen training points: relative percentages errors for different values of the number of training samples.

data $\bar{u}(x) = \frac{\lVert x\rVert^2}{n}$ and the exact solution given by

$$u(x,t) = \frac{\lVert x\rVert^2}{n} + 2t. \tag{4.16}$$

The observation domain is set as $D'_T = [a, 1-a] \times [a, 1-a]^n$ with $a = 0.4$ and the data term (4.2) is defined by restricting the exact solution (4.16) to this observation doman.

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 n 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 2.3 is run with loss function (4.9) and the best performing hyperparameters are identified after ensemble training and presented in Table 6. The resulting L^2-error $\lVert u-u^*\rVert_{L^2(D_T)}$ (in relative percentages and computed on a test set of 10^5 randomly chosen samples) is shown in Table 6. We observe from this table that the PINN error is very small, less than 0.1% till $n = 10$ space dimensions. From then on, the error increases apparently linearly, with still very low errors of 2% for $n = 100$ 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*, where one would expect an exponential increase of the error with respect to dimension.

- 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.

- This experiment should be compared to experiment 3.4.3 in [45], where the forward problem for the heat equation in very high dimensions was approximated with PINNs. Compared to the results presented there (see Table 6 in [45]), 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 number of training samples*, than for the forward problem. This surprising observation merits further investigation and highlights the potential of PINNs in solving inverse problems.

n	N	$K-1$	$\tilde{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\lvert\lvert u-u^*\rvert\rvert_{L^2}$
1	16384	4	20	0.0	0.01	$5.1\cdot 10^{-5}$	0.012%
5	16384	4	20	0.0	0.01	0.0003	0.044%
10	16384	4	20	0.0	0.001	0.0003	0.07%
20	16384	4	20	10^{-6}	0.001	0.0035	0.62%
50	16384	4	20	10^{-6}	0.001	0.0038	1.68%
100	16384	4	20	10^{-6}	0.001	0.003	2.0%

Table 6: Data assimilation problem for the Multi (n-D) dimensional Heat equation with randomly chosen training points: relative percentages errors for different number of dimensions.

5 The Wave equation

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

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,

$$
\begin{aligned}
u_{tt} - \Delta u = f, \quad & \forall (x, t) \in D \times (0, T), \\
u \equiv 0, \quad & \forall (x, t) \in \partial D \times (0, T),
\end{aligned}
\tag{5.1}
$$

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,

$$
\begin{aligned}
u(x, 0) = u_0(x), \quad & \forall x \in D, \\
u_t(x, 0) = u_1(x), \quad & \forall x \in D,
\end{aligned}
\tag{5.2}
$$

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,

$$
u(x, t) = g, \quad (x, t) \in D' \times (0, T),
\tag{5.3}
$$

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 (5.1), given data f, g. A slight variant of this problem stems from photoacoustic tomography (PAT) [28] and it also arises in control theory, in the form of so-called Luenberger observers [52].

The data assimilation problem for the wave equation (5.1), (5.3) 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 [2], is satified,

Definition 5.1. Geometric Control Condition*(GCC) (see [2, 54]): 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 [2, 54] for the rather technical definition of generalized bi-characteristics. 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 [43] roughly requires that the final time T and the set $\partial D' \cap \partial D$ are relatively large.

Under the GCC, one has follow [9] and prove the following well-posedness result for the data assimilation problem for the wave equation (5.1),(5.3),

Theorem 5.2. *[[9], 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,*

$$
\sup_{t \in [0,T]} \left(\|u(\cdot, t)\|_{L^2(D)} + \|\partial_t u(\cdot, t)\|_{H^{-1}(D)} \right) \leqslant \kappa \left(\|u\|_{L^2(D'_T)} + \|u_{tt} - \Delta u\|_{L^2(D_T)} + \|u\|_{L^2(\partial D \times (0,T))} \right),
\tag{5.4}
$$

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 [2], which are derived by using micro-local propagation of singularities for the wave equation. Alternative proofs use Carleman estimates [4].

Again, we can recast the data assimilation inverse problem for the wave equation in the abstract formulation of section 2, with (5.4) playing the role of the conditional stability estimate (2.6). Thus, this inverse problem is amenable to efficient approximation by PINNs.

5.2 PINNs

We specify the algorithm 2.3 to generate a PINN for approximating the data assimilation inverse problem (5.1), (5.3), in the following steps,

5.2.1 Training sets

We consider exactly the same training sets as for the heat equationn i.e, $\mathcal{S}_d = \{z_j\}$ for $z_j = (x,t)_j \in D_T'$, with $1 \leqslant j \leqslant N_d$, are quadrature points, corresponding to the quadrature rule (2.10), $\mathcal{S}_{int}$ as set of quadrature points $y_i = (x,t)_i \in D_T$, for $1 \leqslant i \leqslant N_{int}$, corresponding to the quadrature rule (2.8) and *spatial boundary* training set $\mathcal{S}_{sb} = \{\bar{y}_i\}$ for $1 \leqslant i \leqslant N_{sb}$, with $\bar{y}_i = (\bar{x},t)_i$, 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* (4.4).

5.2.2 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 $k \geqslant 2$. We define the following residuals that are needed in algorithm 2.3. The PDE residual (2.16) is given by,

$$\mathcal{R}_\theta = \partial_{tt}u_\theta - \Delta u_\theta - f, \quad \forall (x,t) \in D_T. \tag{5.5}$$

We need the following residual to account for the boundary data in (5.1),

$$\mathcal{R}_{sb,\theta} = u_\theta|_{\partial D \times (0,T)}. \tag{5.6}$$

The data residual (2.17) is given by,

$$\mathcal{R}_{d,\theta} = u_\theta - g, \quad \forall (x,t) \in D_T'. \tag{5.7}$$

5.2.3 Loss functions

In algorithm 2.3 for approximating the inverse problem (5.1), (5.3), we will need the following loss function (which is identically the same as the loss function (4.9) for the heat equation),

$$J(\theta) = \sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta}(z_j)|^2 + \sum_{i=1}^{N_{sb}} w_i^{sb} |\mathcal{R}_{sb,\theta}(\bar{y}_i)|^2 + \lambda \sum_{i=1}^{N_{int}} w_i |\mathcal{R}_\theta(y_i)|^2, \tag{5.8}$$

with hyperparamter λ, residuals defined in (5.7), (5.5) and (5.6), training points defined above and weights w_j^d, w_i and w_i^{sb}, corresponding to quadrature rules (2.10), (2.8) and (4.4), respectively.

5.3 Estimates on the generalization error

Denoting $u^* = u_{\theta^*}$ as the PINN, generated by algorithm 2.3 to approximate the data assimilation problem (5.1), (5.3), for the wave equation, we define the following generalization error,

$$\mathcal{E}_G := \|u - u^*\|_{C^1([0,T];H^{-1}(D)) \cap C([0,T];L^2(D))} = \sup_{t \in [0,T]} \left(\|u(\cdot,t) - u^*(\cdot,t)\|_{L^2(D)} + \|\partial_t u(\cdot,t) - u^*(\cdot,t)\|_{H^{-1}(D)} \right). \tag{5.9}$$

As in theorem 2.4 and the previous examples, we will bound the generalization error in terms of the following training errors,

$$\mathcal{E}_{d,T} = \left(\sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta^*}(z_j)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_{int,T} = \left(\sum_{i=1}^{N_{int}} w_i |\mathcal{R}_{\theta^*}(y_i)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_{sb,T} = \left(\sum_{i=1}^{N_{sb}} w_i^{sb} |\mathcal{R}_{\theta^*}(\bar{y}_i)|^2 \right)^{\frac{1}{2}}. \tag{5.10}$$

The training errors $\mathcal{E}_{int,T}, \mathcal{E}_{sb,T}$ and $\mathcal{E}_{d,T}$, can be readily computed from the loss functions (2.21), (5.8), a posteriori. We have the following estimate on the generalization error in terms of the training error,
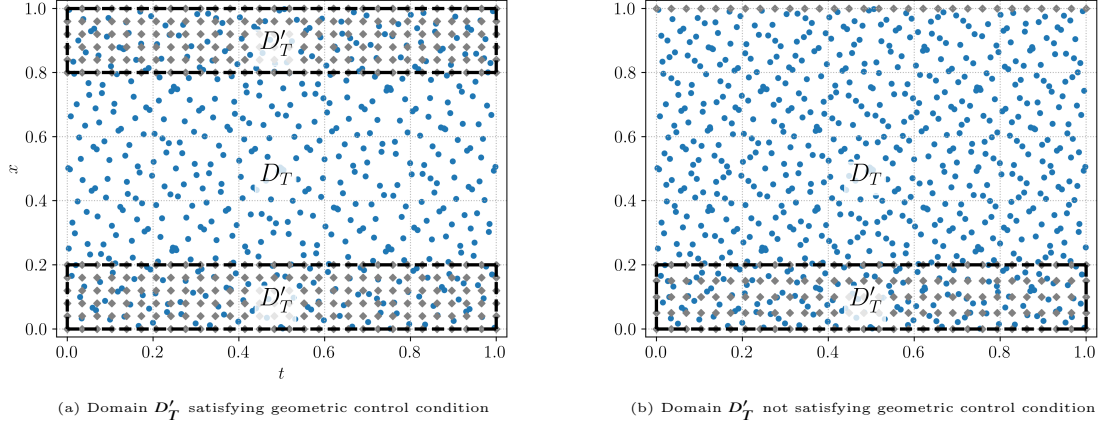
(a) Domain D'_T satisfying geometric control condition

(b) Domain D'_T not satisfying geometric control condition

Figure 6: 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).

Lemma 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 $k \geqslant 2$, 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 (5.1) and satisfies the data (5.3). Let $u^* = u_{\theta^*} \in C^k(D_T)$ be a PINN generated by the algorithm 2.3, with loss functions (2.21), (5.8). Then, the generalization error (5.9) is bounded by,*

$$\mathcal{E}_G \leqslant \kappa \left(\mathcal{E}_{d,T} + \mathcal{E}_{int,T} + \mathcal{E}_{sb,T} + C_q^{\frac{1}{2}} N_{int}^{-\frac{\alpha}{2}} + C_{bd}^{\frac{1}{2}} N_{sb}^{-\frac{\alpha_{sb}}{2}} + C_{qd}^{\frac{1}{2}} N_d^{-\frac{\alpha_d}{2}} \right), \tag{5.11}$$

for observability constant κ depending on the underlying geometry, T, u, u^* and with constants $C_q = C_q \left(\|\mathcal{R}_{\theta^*}\|_{C^{k-2}(D_T)} \right), C_{bd} = C_{bd} \left(\|\mathcal{R}_{sb,\theta^*}\|_{C^k(\partial D \times (0,T))} \right)$ and $C_{qd} = C_{qd} \left(\|\mathcal{R}_{d,\theta^*}\|_{C^k(D'_T)} \right)$, given by the quadrature error bounds (2.9), (4.5) and (2.11), respectively.

Proof. For notational simplicity, we denote $\mathcal{R} = \mathcal{R}_{\theta^*}, \mathcal{R}_{sb} = \mathcal{R}_{sb,\theta^*}$ and $\mathcal{R}_d = \mathcal{R}_{d,\theta^*}$.

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 (5.1) and the data observable in (5.3) and by definitions (5.5),(5.6),(4.8), we see that $\hat{u}$ satisfies,

$$\hat{u}_{tt} - \Delta \hat{u} = \mathcal{R}, \quad \forall (x,t) \in D_T,$$
$$\hat{u}|_{\partial D \times (0,T)} = \mathcal{R}_{sb}, \tag{5.12}$$
$$\hat{u} = \mathcal{R}_d, \quad \forall (x,t) \in D'_T.$$

Therefore, we can apply the observability estimate (5.4) to obtain,

$$\sup_{t \in [0,T]} \left(\|\hat{u}(\cdot,t)\|_{L^2(D)} + \|\partial_t \hat{u}(\cdot,t)\|_{H^{-1}(D)} \right) \leqslant \kappa \left(\|\mathcal{R}_d\|_{L^2(D'_T)} + \|\mathcal{R}_{int}\|_{L^2(D_T)} + \|\mathcal{R}_{sb}\|_{L^2(\partial D \times (0,T))} \right), \tag{5.13}$$

with the observability constant κ from (5.4).

Realizing that the training errors $\mathcal{E}_{d,T}^2$, $\mathcal{E}_{int,T}^2$ and $\mathcal{E}_{sb,T}^2$ are the quadrature approximations for $\|\mathcal{R}_d\|_{L^2(D')_T}^2$, $\|\mathcal{R}_{int}\|_{L^2(D_T)}^2$ and $\|\mathcal{R}_{sb}\|_{L^2(\partial D \times (0,T))}^2$ with respect to the quadrature rules (2.10), (2.8) and (4.4), respectively and using bounds (2.11), (2.9) and (4.5) and substituting the result into (5.13) yields the desired bound (5.11). □

5.4 Numerical experiments

We present the following experiment proposed in [9], 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

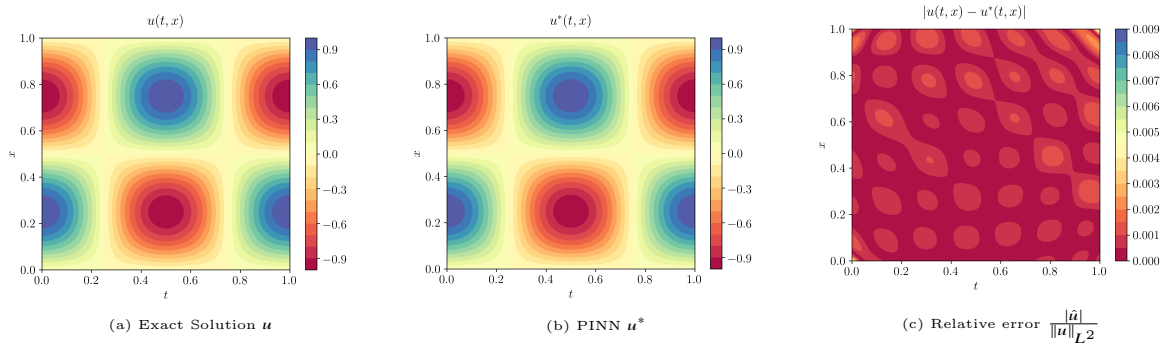(a) Exact Solution u (b) PINN u^* (c) Relative error $\frac{|\tilde{u}|}{\|u\|_{L^2}}$

Figure 7: Data assimilation problem for the Wave equation in domains shown in figure 6 (left) satisfying the geometric control condition. Exact solution, PINN with $N = 60 \times 60$ training points and error.

N	$K-1$	$\tilde{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\|u - u^*\|_{L^2}$
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 7: 1-D Wave equation with observation domain satisfying the geometric control condition and shown in figure 6(left): relative percentage L^2 errors for different values of the number of training samples.

term g in (5.3) are generated from the exact solution

$$u(t, x) = \sin(2\pi t) \sin(2\pi x), \tag{5.14}$$

resulting in $f = 0$ and also satisfying the zero Dirichlet boundary conditions of (5.1).

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 6 for an illustration of the domains. Note that the resulting observation domain D'_T satisfies the geometric control condition [9]. 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 7, 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 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 7, where we present the errors for a sequence of PINNs (with increasing number of training points). We focus on the $\sup_{t \in [0,T]} \left(\|u(\cdot, t) - u(\cdot, t)^*\|_{L^2(D)} \right)$ error, which can be readily bounded above by the generalization error bound (5.11). 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 [9]. The interior, spatial boundary and data training points are chosen similarly to the previous numerical experiment and are illustrated in figure 6 (right). In figure 8, we plot the exact solution and the PINN (and the error), corresponding to $N = N_{int} + N_{sb} + N_d = 60 \times 60$, 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 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 [9], the errors are of small amplitude, even for this example. This is further verified in Table 8 where we present the generalization errors for this example for a sequence of PINNs, with

(a) Exact Solution u

(b) PINN u^*

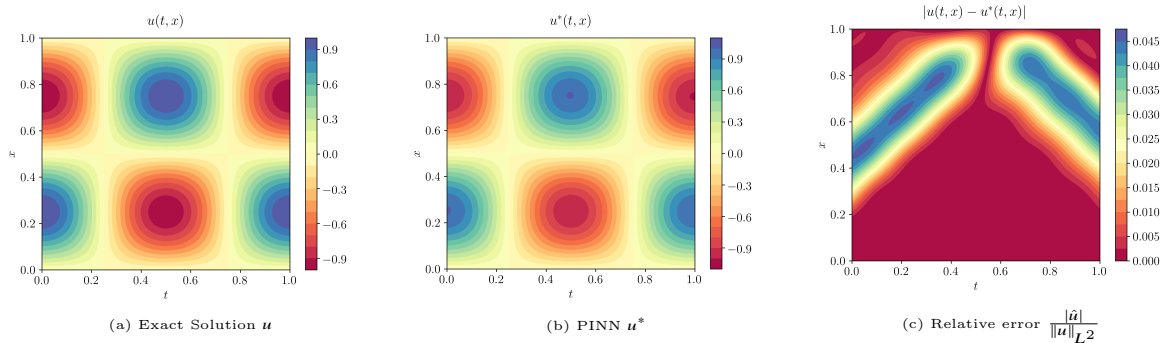(c) Relative error $\frac{|\tilde{u}|}{\|u\|_{L^2}}$

Figure 8: Data assimilation problem for the Wave equation in domains shown in figure 6 (right) satisfying the geometric control condition. Exact solution, PINN with $N = 60 \times 60$ training points and error.

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 [2] in this context.

N	$K-1$	$\tilde{d}$	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 8: 1-D Wave equation with observation domain not satisfying the geometric control condition and shown in figure 6 (right): relative percentage L^2 errors for different values of the number of training samples.

6 The Stokes equation

The effectiveness of PINNs in approximating inverse problems was brilliantly showcased in the recent paper [51], 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.

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:

$$
\begin{aligned}
\Delta \mathbf{u} + \nabla p &= \mathbf{f}, \quad \forall x \in D, \\
\operatorname{div}(\mathbf{u}) &= f_d, \quad \forall x \in D.
\end{aligned}
\tag{6.1}
$$

Here, $\mathbf{u} : D \mapsto \mathbb{R}^d$ is the velocity field, $p : D \mapsto \mathbb{R}$ is the pressure and $\mathbf{f} : D \mapsto \mathbb{R}^d$, $f_d : D \mapsto \mathbb{R}$ are source terms.

Note that the Stokes equation (6.1) is not well-posed as we are not providing any boundary conditions. In the corresponding data assimilation problem [10] and references therein, one provides the following *data*,

$$
\mathbf{u} = \mathbf{g}, \quad \forall x \in D',
\tag{6.2}
$$

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 $\mathbf{u}$ (and the pressure p)), given $\mathbf{f}, f_d$ and $\mathbf{g}$. In particular,

we wish to find solutions $\mathbf{u} \in H^1(D; \mathbb{R}^d)$ and $p \in L_0^2(D)$ (i.e, square integrable functions with zero mean), such that the following holds,

$$\int_D \nabla \mathbf{u} \cdot \nabla \mathbf{v} dx + \int_D p \operatorname{div}(\mathbf{v}) dx = \int_D \mathbf{f} \mathbf{v} dx,$$

$$\int_D \operatorname{div}(\mathbf{u}) w dx = \int_D f_d w dx, \tag{6.3}$$

for all test functions $\mathbf{v} \in H_0^1(D; \mathbb{R}^d)$ and $w \in L^2(D)$.

The well-posedness and conditional stability estimates for the data assimilation problem for the Stokes equation (6.1), (6.2) has been extensively investigated in [34] and references therein. In particular, we have the following stability estimate,

Theorem 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 $\mathbf{u} \in H^1(D; \mathbb{R}^d)$ and $p \in L_0^2(D)$ satisfy (6.3) for all test functions $\mathbf{v} \in H_0^1(D; \mathbb{R}^d)$ and $w \in L^2(D)$. Then for any $\mathbf{f} \in L^2(D; \mathbb{R}^d)$, $f_d \in L^2(D)$ and for any $R_2 > R_1$ such that $B_{R_2}(x_0) \subset D$, the following bound holds,*

$$\|\mathbf{u}\|_{L^2(B_{R_2}(x_0))}^2 \leqslant C \left(\mathcal{C}_{\mathbf{f},f_d} + \mathcal{C}_{\mathbf{u}}^{\tau} \mathcal{C}_{\mathbf{f},f_d}^{1-\tau} + \left(\mathcal{C}_{\mathbf{f},f_d}^{1-\tau} \mathcal{C}_{\mathbf{u}}^{1-\tau} \right) \|\mathbf{u}\|_{L^2(D')}^{2\tau} \right), \tag{6.4}$$

with constants defined as,

$$\mathcal{C}_{\mathbf{f},f_d} := \|\mathbf{f}\|_{L^2(D)}^2 + \|f_d\|_{L^2(D)}^2, \quad \mathcal{C}_{\mathbf{u}} := \|\mathbf{u}\|_{L^2(D)}^2. \tag{6.5}$$

Proof. Let $\mathbf{u}_1, p_1$ be solutions of the following boundary-value problem for the Stokes equations,

$$\begin{aligned} \Delta \mathbf{u}_1 + \nabla p_1 &= \mathbf{f}, \quad \forall x \in D, \\ \operatorname{div}(\mathbf{u}_1) &= f_d, \quad \forall x \in D, \\ \mathbf{u}_1 &\equiv 0, \quad \forall x \in \partial D. \end{aligned} \tag{6.6}$$

We know from classical theory for the Stokes' equation [55], [10](estimate 2.2) that,

$$\|\mathbf{u}_1\|_{H^1(D)} + \|p_1\|_{L^2(D)} \leqslant C \left(\|\mathbf{f}\|_{L^2(D)} + \|f_d\|_{L^2(D)} \right), \tag{6.7}$$

for some constant C that depends only on the domain D.

Define $\mathbf{u}_2, p_2$ as the solutions of the following homogeneous Stokes' equations,

$$\begin{aligned} \Delta \mathbf{u}_2 + \nabla p_2 &= 0, \quad \forall x \in D, \\ \operatorname{div}(\mathbf{u}_2) &= 0, \quad \forall x \in D. \end{aligned} \tag{6.8}$$

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 [34] holds for the solutions $\mathbf{u}_2 \in H^1(D; \mathbb{R}^d)$ and $p \in H^1(D)$ of the homogeneous Stokes equations (6.8),

$$\int_{B_{R_2}(x_0)} |\mathbf{u}_2|^2 dx \leqslant C \left(\int_{B_{R_1}(x_0)} |\mathbf{u}_2|^2 dx \right)^{\tau} \left(\int_{B_{R_3}(x_0)} |\mathbf{u}_2|^2 dx \right)^{1-\tau}, \tag{6.9}$$

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 [34] for more on the constants).

It is easy to check that $\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2$ and $p = p_1 + p_2$, satisfy the Stokes equation (6.1) in the weak sense, i.e they satisfy (6.3). By identifying all constants in the following calculation with a generic constant C,

we have,

$$\int_{B_{R_2}(x_0)} |\mathbf{u}|^2 dx = \int_{B_{R_2}(x_0)} |\mathbf{u}_1 + \mathbf{u}_2|^2 dx$$

$$\leqslant C \left(\int_{B_{R_2}(x_0)} |\mathbf{u}_1|^2 dx + \int_{B_{R_2}(x_0)} |\mathbf{u}_2|^2 dx \right) \leqslant C \left(\int_D |\mathbf{u}_1|^2 dx + \int_{B_{R_2}(x_0)} |\mathbf{u}_2|^2 dx \right)$$

$$\leqslant C \left(\|\mathbf{f}\|_{L^2(D)}^2 + \|f_d\|_{L^2(D)}^2 + \left(\int_{B_{R_1}(x_0)} |\mathbf{u}_2|^2 dx \right)^\tau \left(\int_{B_{R_3}(x_0)} |\mathbf{u}_2|^2 dx \right)^{1-\tau} \right) \quad \text{(by (6.7), (6.9))}$$

$$\leqslant C \left(\|\mathbf{f}\|_{L^2(D)}^2 + \|f_d\|_{L^2(D)}^2 + \left(\int_{B_{R_1}(x_0)} |\mathbf{u} - \mathbf{u}_1|^2 dx \right)^\tau \left(\int_{B_{R_3}(x_0)} |\mathbf{u} - \mathbf{u}_1|^2 dx \right)^{1-\tau} \right)$$

$$\leqslant C \left(\|\mathbf{f}\|_{L^2(D)}^2 + \|f_d\|_{L^2(D)}^2 + \|\mathbf{u}\|_{L^2(B_{R_1}(x_0))}^{2\tau} \|\mathbf{u}\|_{L^2(D)}^{2(1-\tau)} \right)$$

$$+ C \left(\|\mathbf{u}\|_{L^2(B_{R_1}(x_0))}^{2\tau} \|\mathbf{u}_1\|_{L^2(D)}^{2(1-\tau)} + \|\mathbf{u}_1\|_{L^2(D)}^{2\tau} \|\mathbf{u}\|_{L^2(D)}^{2(1-\tau)} + \|\mathbf{u}_1\|_{L^2(D)}^2 \right).$$

Substituting (6.7) and identifying the constants yields the desired bound (6.4).

$\square$

As in the previous examples, this inverse problem can be recast in terms of the abstract formalism of section 2, with the conditional stability estimate (6.4) occupying the role of (2.6) and making this inverse problem amenable to efficient approximation by PINNs.

6.2 PINNs

We specify the algorithm 2.3 to generate a PINN for approximating the inverse problem (6.1), (6.2) in the following steps,

6.2.1 Training sets

As the training set $\mathcal{S}_{int}$ in algorithm 2.3, we take a set of quadrature points $y_i \in D$, for $1 \leqslant i \leqslant N_{int}$, corresponding to the quadrature rule (2.8). 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 $\mathcal{S}_d = \{z_j\}$ for $z_j \in D'$, with $1 \leqslant j \leqslant N_d$, are quadrature points, corresponding to the quadrature rule (2.10).

6.2.2 Residuals

We will require that for parameters $\theta \in \Theta$, the neural networks $\mathbf{u}_\theta \in C^k(D; \mathbb{R}^d), p_\theta \in C^k(D; \mathbb{R})$, for $k \geqslant 2$. We define the following residuals that are needed in algorithm 2.3. The PDE residual (2.16), consists of two parts in this case (see also [45], section 5), given by,

$$\mathcal{R}_{\theta,\mathbf{u}} = \Delta \mathbf{u}_\theta + \nabla p_\theta - \mathbf{f}, \quad \forall x \in D, \tag{6.10}$$

and

$$\mathcal{R}_{\theta,div} = \text{div}(\mathbf{u}_\theta) - \mathbf{g}, \quad \forall x \in D, \tag{6.11}$$

The data residual is given by,

$$\mathcal{R}_{d,\theta} = \mathbf{u}_\theta - \mathbf{g}, \quad \forall x \in D'. \tag{6.12}$$

6.2.3 Loss functions

In algorithm 2.3 for approximating the inverse problem (6.1), (6.2), we will need the following loss function,

$$J(\theta) = \sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta}(z_j)|^2 + \lambda \left(\sum_{i=1}^{N_{int}} w_i |\mathcal{R}_{\mathbf{u},\theta}(y_i)|^2 + \sum_{i=1}^{N_{int}} w_i |\mathcal{R}_{div,\theta}(y_i)|^2 \right), \tag{6.13}$$

with hyperparamter λ, residuals defined in (3.7), (3.8), training points defined above and weights w_i, w_j^d, corresponding to quadrature rules (2.8) and (2.10), respectively.
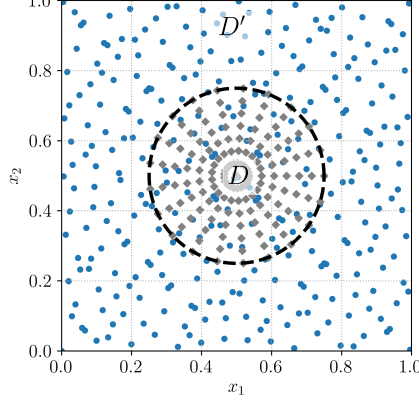

Figure 9: The domains D, D' for the numerical experiment for the Stokes equation. Training set $\mathcal{S}_{int}$ are Sobol points (blue dots) and training set $\mathcal{S}_d$ are Radial (Cartesian) grid points (grey squares).

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,

$$\mathcal{E}_G(B_R) := \|\mathbf{u} - \mathbf{u}^*\|_{L^2(B_R(x_0))}, \tag{6.14}$$

with $\mathbf{u}^* = \mathbf{u}_{\theta^*}$ is the PINN generated by algorithm 2.3. We will estimate this generalization error in terms of the training errors defined by,

$$\mathcal{E}_{d,T} = \left(\sum_{j=1}^{N_d} w_j^d |\mathcal{R}_{d,\theta^*}(z_j)|^2 \right)^{\frac{1}{2}}, \quad \mathcal{E}_{p,T} = \left(\sum_{i=1}^{N_{int}} w_i |\mathcal{R}_{\mathbf{u},\theta^*}(y_i)|^2 + \sum_{i=1}^{N_{int}} w_i |\mathcal{R}_{div,\theta}(y_i)|^2 \right)^{\frac{1}{2}}, \tag{6.15}$$

in the following lemma,

Lemma 6.2. *For $\mathbf{f} \in C^{k-2}(D; \mathbb{R}^d)$, $f_d \in C^{k-1}(D)$ and $\mathbf{g} \in C^k(D')$, with $k \geqslant 2$. Let $\mathbf{u} \in H^1(D; \mathbb{R}^d)$ and $p \in H^1(D)$ be the solution of the inverse problem corresponding to the Stokes' equations (6.1) i.e, they satisfy (6.3) for all test functions $\mathbf{v} \in H_0^1(D; \mathbb{R}^d)$, $w \in L^2(D)$ and satisfies the data (6.2). Let $\mathbf{u}^* = \mathbf{u}_{\theta^*} \in C^k(D; \mathbb{R}^d), p^* = p_{\theta^*} \in C^k(D)$ be a PINN generated by the algorithm 2.3, with loss functions (2.21), (6.13). Let $B_{R_1}(x_0)$ be the largest ball inside $D' \subset D$. Then, the generalization error (6.14) for balls $B_R(x_0) \in D$ with $R > R_1$ is bounded by,*

$$(\mathcal{E}_G(B_R))^2 \leqslant$$

$$C \left[\mathcal{E}_{p,T}^2 + \mathcal{C}_{\hat{\mathbf{u}}}^\tau \mathcal{E}_{p,T}^{2(1-\tau)} + \mathcal{C}_{\hat{\mathbf{u}}}^{1-\tau} \mathcal{E}_{p,T}^{2(1-\tau)} \mathcal{E}_{d,T}^{2\tau} + C_q N^{-\alpha} + \mathcal{C}_{\hat{\mathbf{u}}}^\tau C_q^{(1-\tau)} N^{-(1-\tau)\alpha} + \mathcal{C}_{\hat{\mathbf{u}}}^{1-\tau} C_q^{(1-\tau)} C_{qd}^\tau N^{-(1-\tau)\alpha} N_d^{-\alpha_d \tau} \right] \tag{6.16}$$

with constants $\mathcal{C}_{\hat{\mathbf{u}}} = \|\mathbf{u}\|_{L^2(D)}^2 + \|\mathbf{u}^*\|_{L^2(D)}^2$, $C_q = C_q \left(\|\mathcal{R}_{\mathbf{u},\theta^*}\|_{C^{k-2}(D)} + \|\mathcal{R}_{div,\theta^*}\|_{C^{k-1}(D)} \right)$ and $C_{qd} = C_{qd} \left(\|\mathcal{R}_{d,\theta^*}\|_{C^k(D')} \right)$, given by the quadrature error bounds (2.9) and (2.11) and τ given in (6.4).

Proof. For notational simplicity, we denote $\mathcal{R}_{\mathbf{u}} = \mathcal{R}_{\mathbf{u},\theta^*}, \mathcal{R}_{div} = \mathcal{R}_{div,\theta^*}$ and $\mathcal{R}_d = \mathcal{R}_{d,\theta^*}$.

Define $\hat{\mathbf{u}} = \mathbf{u}^* - \mathbf{u} \in H^1(D; \mathbb{R}^d)$ and $\hat{p} = p^* - p \in H^1(D)$, by linearity of the differential operator in (6.1) and the data observable in (6.2) and by definitions (6.10), (6.11), (6.12), we see that $\hat{\mathbf{u}}, \hat{p}$ satisfy,

$$\begin{aligned} \Delta \hat{\mathbf{u}} + \nabla \hat{p} &= \mathcal{R}_{\mathbf{u}}, \quad \forall x \in D, \\ \mathrm{div}(\hat{\mathbf{u}}) &= \mathcal{R}_{div}, \quad \forall x \in D, \\ \hat{\mathbf{u}} &= \mathcal{R}_d, \quad \forall x \in D', \end{aligned} \tag{6.17}$$

28

with Stokes' equation being satisfied in the sense of (6.3).

Hence, we can directly apply the conditional stability estimate (6.4) to obtain,

$$\mathcal{E}_G^2(B_R) \leqslant C \left(\mathcal{C}_\mathcal{R} + \mathcal{C}_{\hat{\mathbf{u}}}^\tau \mathcal{C}_\mathcal{R}^{1-\tau} + \left(\mathcal{C}_\mathcal{R}^{1-\tau} \mathcal{C}_{\hat{\mathbf{u}}}^{1-\tau} \right) \|\mathcal{R}_d\|_{L^2(D')}^{2\tau} \right), \tag{6.18}$$

with constants,

$$\mathcal{C}_\mathcal{R} = \|\mathcal{R}_\mathbf{u}\|_{L^2(D)}^2 + \|\mathcal{R}_{div}\|_{L^2(D)}^2, \quad \mathcal{C}_{\hat{\mathbf{u}}} = \|\mathbf{u}\|_{L^2(D)}^2 + \|\mathbf{u}^*\|_{L^2(D)}^2. \tag{6.19}$$

By observing that $\mathcal{E}_{d,T}^2$ is the quadrature approximation of $\|\mathcal{R}_d\|_{L^2(D')}^2$ with the quadrature rule (2.10) and $\mathcal{E}_{p,T}^2$ is the quadrature approximation of $\mathcal{C}_\mathcal{R}$ with the quadrature ruel (2.8), we can directly apply (2.9) and (2.11) to obtain the desired inequality (6.16). □

Remark 6.3. The bound (6.16) 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 $\mathcal{E}_{p,T}, \mathcal{E}_{d,T}$, 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 6.4. We observe that the generalization error in (6.16) 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 (6.14) 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. ∎
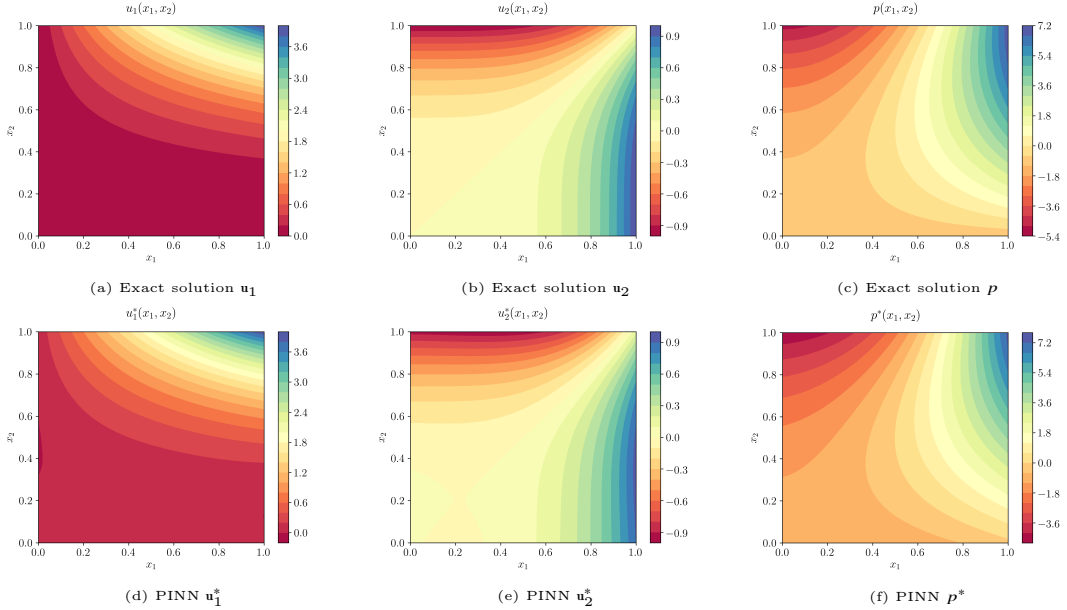

Figure 10: Comparison of the Exact solution and the PINN approximations for the inverse problem for the Stokes equation, with $N = 20 \times 20$ training points.

6.4 Numerical experiments

We follow [10] and consider the homogeneous version of Stokes equation i.e $\mathbf{f}, f_d \equiv 0$, in the two-dimensional domain $(0,1)^2$. We consider the exact solutions,

$$\mathbf{u}(x_1, x_2) = (4x_1 x_2^3, x_1^4 - x_2^4), \quad p(x_1, x_2) = 12x_1^2 x_2 - 4x_2^3 - 1, \tag{6.20}$$

and define the data term $\mathbf{g} = \mathbf{u}|_{D'}$ on the sub-domain,

$$D' = \{(x_1, x_2) \in \mathbb{R}^2 : \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} < 0.25\}. \tag{6.21}$$

The domain D and the sub-domain D' are illustrated in figure 9. 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 9 for an illustration of these training sets.

With these training sets, we run algorithm 2.3 with loss function (2.21), (6.13) to obtain PINNs approximating the data assimilation problem for the Stokes equation (6.1), (6.2). The results with a PINN, trained with $N = 40 \times 40$ and $N_d = rN$ points, with $r = \text{meas}(D')/\text{meas}(D)$, are shown in figure 10, where we plot the exact and PINNs solutions for the velocity components $\mathbf{u}_1$, $\mathbf{u}_2$ 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 9, we present the training errors and relative percentage errors of $\|\mathbf{u} - \mathbf{u}^*\|_{L^2(D)}$ 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 (6.16) on the generalization error. More surprisingly, we also observe from the Table 9 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 (6.2). 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 (6.12), and the loss function (6.13) 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 9) 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	$K-1$	$\tilde{d}$	λ_{reg}	λ	$\mathcal{E}_T$	$\|\mathbf{u} - \mathbf{u}^*\|_{L^2}$	$\|p - p^*\|_{L^2}$
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 9: Stokes equation: Relative percentages errors for different numbers of training points.

Code

The building and the training of PINNs, together with the ensemble training for the selection of the model hyperparameters, are performed with a collection of Python scripts, realized with the support of PyTorch `https://pytorch.org/`. The scripts can be downloaded from `https://github.com/mroberto166/Pinns`.

7 Discussion

Inverse problems for PDEs are very challenging computationally on account of the difficulties of finding stable regularizations and possibly high computational cost. Physics informed neural networks have recently been very successfully applied to efficiently simulate the solutions of several inverse problems for (nonlinear) PDEs, [42, 50, 51] and references therein. However, rigorous justifications are currently lacking.

Our main aim in this paper was to provide a rigorous justification for using PINNs to approximate the solutions of a large class of inverse problems for PDEs. To this end, we focussed on the so-called *data assimilation* or *unique continuation* inverse problems in this paper. For these problems, the inputs necessary for the forward problem of the underlying PDE (for instance the abstract pde (2.1)), such as initial and boundary conditions are unknown. However, one has access to (possibly noisy) observables of the solution ((2.4) for the abstract pde (2.1)), on the so-called *observation domain*, which a subset of the underlying domain. Such data assimilation problems arise in many applications, particularly in geophysics and meteorology.

In section 2, we posed the data assimilation inverse problem for the abstract PDE (2.1), (2.4). The corresponding algorithm 2.3 to generate a PINN for approximating this inverse problem was proposed. Key ingredients in this algorithm included using quadrature points, corresponding to underlying quadrature rules (2.8) as *training points*, and the (strong form of) PDE residual (2.16), collocated at these training points as the loss function (2.20), for training the neural network. Under the assumption that solutions to the underlying inverse problem (2.1), (2.4), satisfy a *conditional stability estimate* (2.6), we were able to prove a rigorous upper bound on the generalization (approximation) error (2.22) of the PINN, in terms of the (computable) training error (2.23) and the number of training samples (with a rate prescribed by the quadrature errors (2.9),(2.11)). Thus, we are able to leverage the conditional stability estimate on the inverse problem for the underlying PDE into a bound on the generalization error for the approximating PINN.

We illustrate this abstract framework for concrete examples for linear PDEs that arise in a wide variety of models. We consider the Poisson, heat, wave and Stokes equations in this paper. 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 validate 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.

It is instructive at this point to compare the results obtained with PINNs to those with other methods. We select stabilized non-confirming finite element methods, such as the ones presented in [9, 10, 12, 13] and references therein. In these papers, the authors also employed conditional stability estimates to obtain rigorous error estimates for the finite element method. Comparing our numerical results to these papers, we notice that the PINNs were comparable in terms of the amplitude of the error, when compared to finite element methods on fine meshes. On the other hand, PINNs are extremely simple to program, with the same level of code complexity for both the forward and the inverse problem [50]. Moreover, PINNs are able to obtain very small errors with a training time of less than $1 - 2$ minutes for most examples presented here. Given their simplicity and efficiency, we believe PINNs constitute a promising alternative to available methods for the data assimilation problem. Another interesting finding was the ability of PINNs to solve inverse problems in very high dimensions. We illustrated this performance for the heat equation in domains upto 100 space dimensions and observed that PINN, based on random training points, were able to approximate the solution of the inverse problem to high accuracy and without incurring the *curse of dimensionality*.

Summarizing, we are able to provide a unified framework and to the best of our knowledge, *the first rigorous justification* for using PINNs in the context of data assimilation inverse problems for PDEs.

The current paper, as well as the general methodology of PINNs, does have a few shortcomings that we point out below,

- The bound (2.24) on the generalization error (2.22) is in terms of the training error. We do not estimate this error here. However, this is standard in machine learning [41], where the training error is computed *a posteriori* (see the numerical results for training errors in respective experiments). The training error stems from the solution of a non-convex optimization problem in very high dimensions and there are no robust estimates on this problem. Once such estimates become available, they can be readily incorporated in the rhs of (2.24). For the time being, the estimate (2.24) should be interpreted as *as long as the PINN is trained well, it generalizes well*. This is indeed borne out by the results of the numerical experiments.

31

- A particular manifestation of this issue with the training error was encountered in some of the numerical experiments, where the training error reached a very low value, even for a few training samples. Adding further samples led to a saturation of the already low generalization error as the training error could not be reduced further. If this happens, one needs to either change the architecture of the neural network, for instance by using residual or convolutional neural networks or change the optimizer or use standard tricks in machine learning such as batch normalization and dropout to further reduce the training and generalization errors.

- Although our abstract formalism was very general and covered nonlinear PDEs, we only considered concrete examples of linear PDEs in this paper. This is largely on account of simplicity and the ready availability of conditional error estimates. On the other hand, some of the striking success of PINNs in the context of inverse problems, has been for nonlinear PDEs such as the Navier-Stokes equations [51] and even the compressible Euler equations [42]. Although some Carleman estimates, leading to conditional stability bounds, are available for the Navier-Stokes equations [26], applying them to derive bounds on the generalization error is very challenging and will be considered in a forthcoming paper. However, extending our results to other linear PDEs such as Maxwell's equation, Helmholtz equation, advection-difusion equations and even the linearized Navier-Stokes equations, where conditional stability estimates of the form (2.6) are available, is relatively straightforward.

- We have focused on the *deterministic data assimilation problem* here. In practice, data in the observation domain (2.4) is *noisy*. For small amplitude noise, the estimates can be readily extended to bound the generalization error in (2.34) and the efficiency of PINNs in this regime is validated in a numerical experiment for the Poisson's equation with noisy data. However, dealing with data, perturbed by larger amplitudes of noise, necessitates a Bayesian framework, such as the one considered in [35]. Deriving rigorous estimates on the generalization error in a Bayesian framework will be considered in a future paper.

Acknowledgements.

References

[1] G. Alessandrini, L. Rondi, E. Rosset, and S. Vessella. The stability of the cauchy problem for elliptic equations. *Inverse problems*, 25(12):47, 2009.

[2] 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.

[3] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inform. Theory.*, 39(3):930–945, 1993.

[4] L. Baudouin and M. D. B. annd S. Ervedoza. Global carleman estimates for waves and applications. *Commun. PDE.*, 38:556–598, 2013.

[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] L. Bourgeois. A mixed formulation of quasi-reversibility to solve the cauchy problem for laplace's equation. *Inverse problems*, 21(3):1087–1104, 2005.

[7] H. J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[8] E. Burman. Error estimates for stabilized finite element methods applied to ill-posed problems. *C. R. Math. Acad. Sci. Paris*, 352(7-8):655–659, 2014.

[9] E. Burman, A. Feizmohammadi, and L. Oksanen. A finite element data assimilation method for the wave equation. *Math. Comp.*, 89(324):1681–1709, 2020.

[10] E. Burman and P. Hansbo. Stabilized nonconfirming finite element methods for data assimilation in incompressible flows. *Math. Comp.*, 87(311):1029–1050, 2018.

[11] E. Burman, P. Hansbo, and M. Larson. Solving ill-posed control problems by stabilized finite element methods: an alternative to tikhonov regularization. *Inverse problems*, 34(3), 2018.

[12] 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.

[13] 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.

[14] R. E. Caflisch. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, 7:1–49, 1998.

[15] Y. Chen, L. Lu, G. E. Karniadakis, and L. D. Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. Preprint, available from arXiv:1912.01085, 2019.

[16] G. Cybenko. Approximations by superpositions of sigmoidal functions. *Approximation theory and its applications.*, 9(3):17–28, 1989.

[17] W. E, J. Han, and A. 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.

[18] 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(363-382):6, 2018.

[19] L. D. F. Regazzoni and A. Quarteroni. Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational physics*, 397, 2019.

[20] R. Fletcher. *Practical methods of optimization.* John Wiley and sons, 1987.

[21] A. V. Fursikov and O. Y. Imanuvilov. *Controllability of evolution equations.* SNU lecture notes 34, 1996.

[22] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning.* MIT press, 2016.

[23] J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[24] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks.*, 2(5):359–366, 1989.

[25] O. Y. Imanuvilov. Controllability of parabolic equations. *Math. Sb.*, 186(6):109–132, 1995.

[26] O. Y. Imanuvilov. Remarks on exact controllability for the navier-stokes equations. *ESIAM: COCV*, 6:39–72, 2001.

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[28] P. Kuchment and L. Kunyansky. Mathematics of thermoacoustic tomography. *European J. Appl. Math.*, 2(19):191–224, 2008.

[29] F. Laakmann and P. Petersen. Efficient approximation of solutions of parametric linear transport equations by reludnns. Preprint, 2019.

[30] I. E. Lagaris, A. Likas, and P. G. D. Neural-network methods for bound- ary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11:1041–1049, 2000.

[31] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[32] R. Lattés and J. L. Lions. *Méthode de quasi-réversibilité et applications*. Dunod, Paris, 1967.

[33] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[34] 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.

[35] Y. Liu, X. Meng, and G. E. Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. Preprint, available from arXiv:2003.06097, 2020.

[36] 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.

[37] K. . O. Lye, S. Mishra, P. Chandrasekhar, and D. Ray. Iterative surrogate model optimization (ismo): An active learning algorithm for pde constrained optimization with deep neural networks. Preprint,, 2020.

[38] K. O. Lye, S. Mishra, and R. Molinaro. A multi-level procedure for enhancing accuracy of machine learning algorithms. *European Journal of Applied Mathematics*, 2020.

[39] K. O. Lye, S. Mishra, and D. Ray. Deep learning observables in computational fluid dynamics. *Journal of Computational Physics*, page 109339, 2020.

[40] F. C. M. Badra and J. Darde. Stability estimates for the navier-stokes equations and applications to inverse problems. *Discrete Contin. Dyn. Syst. Ser. B*, 21(8):2379–2407, 2016.

[41] A. R. M. Mohri and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[42] 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.

[43] L. Miller. Escape function conditions for the observation, control and stabilization of the wave equation. *SIAM J. Control. Opt.*, 41(5):1554–1566, 2002.

[44] S. Mishra. A machine learning framework for data driven acceleration of computations of differential equations. *Mathematics in Engineering*, 1:118, 2019.

[45] S. Mishra and R. Molinaro. Estimates on the generalization error of physics informed neural networks (pinns) for approximating pdes. Preprint, available from ETH SAM Reports, 2020.

[46] S. Mishra and T. K. Rusch. Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. Preprint, available as arXiv:2005.12564, 2020.

[47] A. 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.

[48] G. Pang, L. Lu, and G. E. Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM journal of Scientific computing*, 41:A2603–A2626, 2019.

[49] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[50] 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.

[51] M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data. *arXiv preprint arXiv:1808.04327*, 2018.

[52] 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.

[53] D. Ray and J. S. Hesthaven. An artificial neural network as a troubled-cell indicator. *Journal of Computational Physics*, 367:166–191, 2018.

[54] J. L. 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.

[55] G. Seregin. *Lecture notes on regularity theory for the Navier-Stokes equations*. World Scientific, 2015.

[56] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[57] 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.

[58] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. Springer Verlag, 2002.

[59] D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.