LIESOLVER: PDE-Constrained Learning for IBVPs via Lie Symmetries

René P. Klausen [*1] **Ivan Timofeev** [*1] **Jonas Naujoks** [1] **Johannes Frank**
Thomas Wiegand [1 2 3] **Sebastian Lapuschkin** [1 4] **Wojciech Samek** [1 2 3]

Abstract

Initial-boundary value problems (IBVPs) provide the essential framework for modelling a wide range of phenomena in physics and engineering. We introduce a novel method for efficiently solving IBVPs using Lie symmetries to enforce the associated partial differential equation (PDE) exactly by construction. By leveraging symmetry transformations, our model embeds the underlying physical laws and learns the solution solely from initial and boundary data. Consequently, the boundary loss directly quantifies domain-wide error, enabling rigorous error estimation for well-posed IBVPs. We implement LIESOLVER and demonstrate its application to linear homogeneous PDEs, showing that it outperforms physics-informed neural networks (PINNs) in both speed and accuracy while yielding compact models. Overall, our approach significantly enhances the efficiency and reliability of predictions for PDE-constrained problems.

1. Introduction

The growing availability of data and computational resources has established machine learning (ML) as a powerful tool across scientific disciplines. Despite this great success, many models operate as black boxes, limiting their transparency and reliability. This has driven demand for models that are not only accurate but also transparent, data-efficient, and consistent (Rudin, 2019; Samek et al., 2021). Incorporating domain knowledge addresses several of these challenges (von Rueden et al., 2021). In scientific contexts, partial differential equations (PDEs) — which govern a wide range of phenomena from physics to finance — provide a natural framework for such integration. *Physics-informed neural networks* (PINNs) (Raissi et al., 2019) represent the most prominent architecture in this field and have achieved strong results in diverse domains of application (Karniadakis et al., 2021).

Concretely, models like PINNs aim to find specific solutions of PDEs that satisfy additional initial and/or boundary conditions; this setting is commonly referred to as an *initial-boundary value problem* (IBVP). A typical example is predicting the evolution of a dynamical system from its initial state. These IBVPs appear in nearly every scientific discipline, making the development of reliable and efficient IBVP solvers a central challenge in computational science. While classical numerical methods, such as finite difference and finite element techniques (LeVeque, 2007), can effectively address these problems, their computational cost scales poorly with problem dimensionality. This curse of dimensionality makes high-dimensional PDEs particularly challenging and has motivated the development of meshless machine learning approaches.

There are generally three strategies for embedding physical laws in ML models (Karniadakis et al., 2021): (i) incorporating them in the optimization, (ii) integrating them into the model architecture, or (iii) augmenting data using physics-based principles. While (iii) requires large amounts of costly-to-generate data, methods like PINNs (i) introduce atypical loss terms that may lead to serious failures (Krishnapriyan et al., 2021). We therefore adopt the second approach, leveraging the inherent symmetries of PDEs — namely their Lie symmetries — by integrating them directly into the model structure. This is applicable in principle to any PDE with known Lie symmetries, as these symmetries inherently generate parametrized and expressive solutions. Moreover, using Lie symmetries instead of directly enforcing PDEs results in more stable and reliable model behaviour.

Our method integrates predetermined parametrized Lie symmetries as learnable building blocks, enabling the model to compose symmetry transformations while provably remaining in the solution space of the underlying PDE. Con-

[1]Department of Artificial Intelligence, Fraunhofer Heinrich Hertz Institute, Berlin, Germany [2]Department of Electrical Engineering and Computer Science, Technische Universität Berlin, Germany [3]BIFOLD Berlin Institute for the Foundations of Learning and Data, Berlin, Germany [4]Centre of eXplainable Artificial Intelligence, Technological University Dublin, Ireland. Correspondence to: René P. Klausen <rene.pascal.klausen@hhi.fraunhofer.de>.

Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

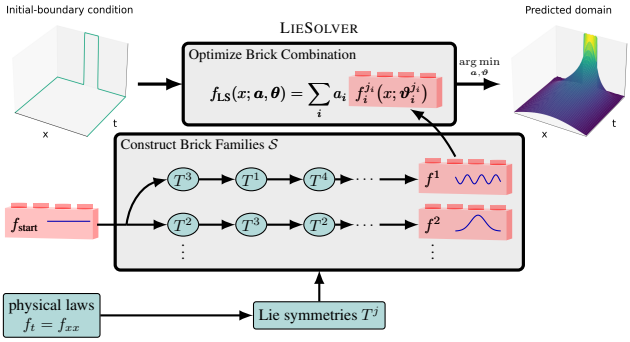


Figure 1. The structure of LIESOLVER, a physics-constrained solver for IBVPs (e.g., time evolution problems). The method consists of two parts. Based on Lie symmetries, one constructs appropriate *bricks* that provably satisfy the physical laws (lower part). This step needs to be performed only once for each physical law (e.g., the heat equation). These bricks are then combined by solving the associated optimization problem to match initial-boundary conditions (upper part). The model provably satisfies the physical laws exactly and approximates the initial-boundary conditions. A detailed description of the pipeline can be found in Section 4.3.

sequently, for any choice of model parameters, the output exactly satisfies the PDE, reducing the optimization problem to fitting the initial-boundary conditions only. This makes optimization significantly more tractable and eliminates the difficult-to-handle PDE loss terms widely considered the principal challenge in PINNs.

As a result, a decrease in the loss function directly corresponds to improved prediction accuracy, making the objective function, unlike in PINNs, a reliable metric for evaluating training progress. Additionally, for well-posed IBVPs, rigorous error bounds can be derived. This approach also enables significantly smaller models (in terms of parameters) while preserving expressiveness, allowing the use of robust second-order optimization schemes that yield more accurate approximations in significantly less time. Currently, LIESOLVER is restricted to linear homogeneous PDEs. Although this class already contains many important problems — such as the wave, heat, advection, convection-diffusion, Klein-Gordon, and Maxwell equations — we are confident that this restriction can be relaxed, since the underlying Lie symmetry principle applies to any PDE.

To summarize the contributions of our work, we provide:

- A novel machine learning framework, LIESOLVER[5] based on Lie symmetries for solving IBVPs. This method (i) enforces PDEs exactly through a PDE-informed architecture, (ii) aligns loss with prediction accuracy, (iii) requires no external training data, as arbitrary data points can be sampled as needed, (iv) allows for rigorous and sharp error bounds, and (v) provides improved interpretability through access to a symbolic decomposition of the solution.
- An implementation[6] of LIESOLVER, with benchmark experiments demonstrating higher solution accuracy and outperforming vanilla PINNs by several orders of magnitude in computation time.

[5]In German, "Lies" is the imperative of "lesen" ("read"); "LieSolver" playfully suggests "Lies Olver" — to read Olver. We use Olver's book (Olver, 1993) as our primary source on Lie symmetries.

2. Related Work

ML Approaches for IBVPs Various methods have been developed over time that employ machine learning techniques to tackle IBVPs. Early approaches to solving differential equations with neural networks (NNs) date back to the pioneering work of (Lagaris et al., 1998); later work such as (Tsoulos & Lagaris, 2006) explored the recovery of analytical closed-form solutions through genetic programming.

The introduction of PINNs (Raissi et al., 2019) revitalized interest in the field of physics-informed machine learning and the application of ML-based techniques for solving PDEs. Subsequent work has focused on improving training stability and scalability using a plethora of approaches, including domain decomposition (Jagtap et al., 2020), adaptive loss balancing (Wang et al., 2022), improved optimizers (Wang et al., 2025; Rathore et al., 2024), resampling (Wu et al., 2023; Lau et al., 2023; Naujoks et al., 2025) and multi-fidelity extensions (Meng et al., 2020). Although promising, PINNs often suffer from unstable convergence (Wang et al., 2021; Krishnapriyan et al., 2021) and practical error estimation and certification remain an ongoing field of research (Eiras et al., 2024; Hillebrecht & Unger, 2025).

Symmetry-Based Approaches In the realm of physics, symmetries play a dominant role (Wigner, 1967), inspiring diverse machine learning approaches which are "symmetry-informed". More generally, this includes, for example, convolutional networks that hard-code translation symmetry (Cohen & Welling, 2016), equivariant networks for three-dimensional particle systems (Thomas et al., 2018), and structure-preserving models such as Hamiltonian (Greydanus et al., 2019) or Lagrangian NNs (Cranmer et al., 2020). These architectures ensure physical consistency by design, but are usually specialized to certain physical aspects.

For PDE-related symmetry approaches more concretely, current research includes symmetry-informed data augmentation for neural PDE solvers (Brandstetter et al., 2022), self-supervised pretraining based on Lie symmetry transformations (Mialon et al., 2023), and symmetry-regularized

[6]The implementation is available here.

PINNs (Akhound-Sadegh et al., 2023), which all exploit the invariance structure of the underlying equations to improve generalization and sample efficiency. Similarly, (Zhang et al., 2023a) leverages Lie symmetry groups to generate data for supervised learning, providing an alternative to physics-informed training. Other methods embed invariant surface conditions into the loss (Zhang et al., 2023b) to enhance the PINN method. Yet another recent line of work (Yang et al., 2024a;b) utilizes Lie symmetries to discover them from data and integrate them into learning architectures, improving long-term predictive accuracy and facilitating practical applications such as equation discovery.

Trefftz-type methods A natural strategy for solving linear homogeneous IBVPs is to approximate the solution by a linear combination of *trial functions* that satisfy the governing PDE exactly by construction, reducing the problem to fitting boundary data alone. This idea underlies the Trefftz method (Trefftz, 1926) and its variants (Kołodziej & Grabski, 2018; Hiptmair et al., 2016), including the method of fundamental solutions (Fairweather & Karageorghis, 1998). The central limitation of these methods is that rich families of such trial functions are known only for a very limited number of PDEs, typically those admitting closed-form solution families such as harmonic polynomials or plane waves. LIESOLVER shares this PDE-exact principle, but overcomes this limitation by generating trial functions — called *bricks* — systematically from the Lie symmetry group of the PDE, making the construction applicable to any PDE with known symmetries. Moreover, the symmetry parameters $\boldsymbol{\vartheta}$ enter each brick nonlinearly, yielding a more expressive approximation than a fixed linear basis — at the cost of a structured nonlinear optimization problem, which LIESOLVER addresses via greedy composition and variable projection.

3. Background

3.1. Initial-Boundary Value Problems (IBVPs)

IBVPs model a wide range of phenomena in science and engineering and are the primary object of study in this work: They couple partial differential equations (PDEs) with initial-boundary conditions (IBCs). Concretely, let $\Omega \subset \mathbb{R}^n$ be an open, bounded, and connected domain and $f : \Omega \to \mathbb{R}^m$ be a sufficiently smooth function. We focus here on *linear homogeneous PDEs*, i.e.

$$\mathcal{D}[f](x) = \sum_{|\alpha| \leq d} c_\alpha(x) \partial^\alpha f(x) = 0 \quad \forall x \in \Omega \,, \tag{1}$$

where the coefficients $c_\alpha(x)$ are given functions and $\alpha \in \mathbb{N}_0^n$ is a multi-index, $|\alpha| := \sum_i \alpha_i$, $\partial^\alpha := \partial_1^{\alpha_1} \cdots \partial_n^{\alpha_n}$. This class covers many key PDEs, such as the wave, heat, advection, convection-diffusion, Klein-Gordon, and Maxwell equations. Any sufficiently smooth function $f : \Omega \to \mathbb{R}^m$ satisfying (1) is called a *solution of the PDE*. In particular, the constant function $f \equiv 0$ is always a solution for any linear homogeneous PDE.

Typically, there are many possible solutions to a PDE. Therefore, we restrict the possible solutions through IBCs. Let $\Gamma_1, \ldots, \Gamma_r \subseteq \partial\Omega$ be smooth connected components of the boundary of Ω. We denote the IBCs as operators

$$\mathcal{B}_k[f](x) = 0 \quad \forall x \in \Gamma_k \;\text{ and }\; k = 1, \ldots, r \;. \tag{2}$$

These operators $\mathcal{B}_k$ usually restrict the function f itself (Dirichlet condition) or its normal derivative (Neumann condition) to certain predefined values, but they can model any kind of (mixed) initial or boundary conditions. Functions $f : \overline{\Omega} \to \mathbb{R}^m$ satisfying (1) and (2) are called *solutions of the IBVP* defined by $\mathcal{D}, \mathcal{B}_1, \ldots, \mathcal{B}_r$.

There are various strategies to solve IBVPs with numerical or symbolic methods. For example, PINNs (Raissi et al., 2019) formulate (1) and (2) as an optimization problem, by training a NN $\phi_\theta(x)$ minimizing the loss terms $\|\mathcal{D}[\phi_\theta](x)\|$ and $\|\mathcal{B}_k[\phi_\theta](x)\|$.

However, not all IBVPs admit a solution. In the following, we therefore always focus on *well-posed* IBVPs, meaning that (i) the IBVP has a solution, (ii) the solution is unique, and (iii) the solution depends continuously on the prescribed initial and boundary data (John, 1978). Further details and references on PDEs and IBVPs are provided in Appendix A.

3.2. Lie Symmetries

PDEs are invariant under certain symmetries, and so are their solutions. These relations are called *Lie symmetries*, which are continuous symmetries among the PDE solutions. Moreover, Lie symmetries are determined solely by the PDE and form parametrized groups (Olver, 1993). Hence, if T_ϑ is such a Lie symmetry transformation of a given PDE depending on a real parameter $\vartheta \in \mathbb{R}$ and f is an arbitrary solution of that PDE, then $T_\vartheta \circ f$ is a solution of the PDE as well. Due to the group property of Lie symmetries, the concatenation of Lie symmetries yields another valid Lie symmetry.

We close this short overview by discussing the Lie symmetries for the one-dimensional heat equation

$$\mathcal{D}[f](x,t) = \frac{\partial f(x,t)}{\partial t} - \frac{\partial^2 f(x,t)}{\partial x^2} = 0 \;, \tag{3}$$

which admits six one-parameter symmetry transformations

(Olver, 1993):

$$T^1_\vartheta : f(x,t) \mapsto f(x-\vartheta,t)\ , \tag{4}$$

$$T^2_\vartheta : f(x,t) \mapsto f(x,t-\vartheta)\ , \tag{5}$$

$$T^3_\vartheta : f(x,t) \mapsto e^{\vartheta} f(x,t)\ , \tag{6}$$

$$T^4_\vartheta : f(x,t) \mapsto f(e^{-\vartheta}x, e^{-2\vartheta}t)\ , \tag{7}$$

$$T^5_\vartheta : f(x,t) \mapsto e^{-\vartheta x+\vartheta^2 t} f(x-2\vartheta t,t)\ , \tag{8}$$

$$T^6_\vartheta : f(x,t) \mapsto e^{-\frac{\vartheta x^2}{1+4\vartheta t}} \frac{f\left(\frac{x}{1+4\vartheta t}, \frac{t}{1+4\vartheta t}\right)}{\sqrt{1+4\vartheta t}}\ . \tag{9}$$

Hence, for any solution $f(x,t)$ of the PDE (3), $T^i_\vartheta \circ f(x,t)$ is another solution of the PDE (3), for any $\vartheta \in \mathbb{R}$ such that $T^i_\vartheta \circ f(x,t)$ is defined. There exists an additional symmetry transformation that reflects the linearity of the PDE

$$T^\alpha_\vartheta : f(x,t) \mapsto f(x,t) + \vartheta\, \alpha(x,t) \tag{10}$$

where $\alpha(x,t)$ is an arbitrary solution of (3). Notably, these symmetries (4)–(10) constitute the complete set of continuous point symmetries of the solution space.

Any PDE possesses its own set of Lie symmetries, which can be determined algorithmically (Hereman, 1996). However, linear homogeneous PDEs always admit symmetries of the form (6) and (10). We provide further details about Lie symmetries and their computation in Appendix B.

4. Methodology of LIESOLVER

4.1. The Structure of LIESOLVER

The key idea of LIESOLVER is to approximate the solution of an IBVP by a linear combination of functions generated by parametrized Lie symmetry transformations (see Figure 1). Since Lie symmetries form a group, they can be concatenated to generate new symmetries. Thus, given an arbitrary solution to start with, one can generate a family of new solutions by successive applications of Lie transformations. Finding such a starting solution is often straightforward. For instance, $f(x) \equiv 1$ solves any linear homogeneous PDE that lacks a zero-order term ($|\alpha| = 0$), e.g., the heat, wave or Laplace equation. We refer to such solutions as *start solutions*, denoted as f_{start}.

By applying a sequence of Lie transformations $T^{l_1}, \dots, T^{l_s}$ to a given f_{start}, we obtain an expressive s-parameter family

$$f(x;\vartheta_1,\dots,\vartheta_s) = T^{l_s}_{\vartheta_s} \cdot \ldots \cdot T^{l_1}_{\vartheta_1} \circ f_{\text{start}}(x)\ , \tag{11}$$

that remains a solution of the underlying PDE for any value of $\boldsymbol{\vartheta} = (\vartheta_1,\dots,\vartheta_s) \in \mathbb{R}^s$, whenever the composed function is defined. We call $f(x;\boldsymbol{\vartheta})$ a *brick solution*. In order to distinguish between symbolic $\boldsymbol{\vartheta}$ and numerical values, we introduce an index for any specific value of $\boldsymbol{\vartheta}$. Thus, $f(x;\boldsymbol{\vartheta})$ represents a *brick family* of solutions, whereas $f_i(x;\boldsymbol{\vartheta}_i)$ is a specific function we call *brick*. Different brick families are distinguished by superscripts $f^j(x;\boldsymbol{\vartheta}^j)$.

Table 1. Comparing purely data-driven NNs with PINNs, LIESOLVER and classical numerical methods FDM, FVM and FEM. See Section 4.4 for discussion and details.

	NN	PINN	LIESOLVER	classical
coverage of well-posed IBVPs	–	non-stiff	linear homogeneous	any
PDE constraint	–	soft	hard	hard
IBC constraint	soft	soft/hard	soft	hard
error estimation	✘	✘	✔	✔
loss alignment	✔	✘	✔	–
respecting physics	✘	✔	✔	✔
meshless/no CoD	✔	✔	✔	✘
simple data fusion	✔	✔	✔	✘
# parameters	●●●	●●○	●○○	–
interpretability	○○○	●○○	●●○	●●●

We consider a set $\mathcal{S} = \{f^1(x;\boldsymbol{\vartheta}^1),\dots,f^N(x;\boldsymbol{\vartheta}^N)\}$ of brick families, where each solution f^j is constructed from a start solution by (11). The algorithm of LIESOLVER chooses and instantiates M bricks from $\mathcal{S}$ (elements can repeat), and combines them linearly:

$$f_{\text{LS}}(x;\boldsymbol{a},\boldsymbol{\theta}) = \sum_{i=1}^{M} a_i\, f_i^{j_i}(x;\boldsymbol{\vartheta}_i^{j_i})\ , \tag{12}$$

where the parameters $\boldsymbol{\theta} = (\boldsymbol{\vartheta}_1^{j_1},\dots,\boldsymbol{\vartheta}_M^{j_M})$ and $\boldsymbol{a} = (a_1,\dots,a_M)$ are specified to certain values. The resulting function f_{LS} is very expressive and covers a major part of the solution space of the PDE (1). Note that each brick $f_i^{j_i}(x;\boldsymbol{\vartheta}_i^{j_i})$ may consist of several parameters as given in (11). We call the collection of bricks appearing in (12) the *active set* $\mathcal{A} = \{f_1^{j_1}(x;\boldsymbol{\vartheta}_1^{j_1}),\dots,f_M^{j_M}(x;\boldsymbol{\vartheta}_M^{j_M})\}$.

Since f_{LS} is a solution of the PDE *by construction* for any $(\boldsymbol{a},\boldsymbol{\theta})$, the problem of finding a solution to the IBVP is reduced to matching given IBCs $\mathcal{B}_1,\dots,\mathcal{B}_r$ (2) by constructing the active set $\mathcal{A}$ and finding optimal parameters $\boldsymbol{a}$ and $\boldsymbol{\theta}$. The set $\mathcal{S}$, built with start solutions and selected transformations applied to them, as well as the number of bricks M are the design choices of the model.

4.2. The LIESOLVER Optimization

The construction of the solution f_{LS} involves determining the optimal active set of bricks $\mathcal{A}$ and optimizing the nonlinear parameters $\boldsymbol{\theta}$ and the linear amplitudes $\boldsymbol{a}$. Unlike traditional deep learning approaches that minimize a single unified loss function via stochastic gradient descent, LIESOLVER employs a multi-stage optimization strategy. It utilizes different objectives for structural learning and parameter tuning. After discussing the data structure, we consider the key components of the LIESOLVER: linear fitting, greedy composition of the active set $\mathcal{A}$ for structural

learning, and nonlinear parameter tuning.

Data Sampling Let $\mathcal{X}_k \subset \Gamma_k$ be finite sets of randomly sampled collocation points from the boundary parts Γ_k according to Section 3.1. The full training set $\mathcal{X} = \bigcup_{k=1}^{r} \mathcal{X}_k \in \mathbb{R}^{L\times n}$ consists of L points $x \in \mathbb{R}^n$. Consequently, the IBCs $\mathcal{B}_1, \dots, \mathcal{B}_r$ define a set of target values $\mathcal{Y} \in \mathbb{R}^{L\times m}$ at the datapoints $\mathcal{X}$.

For the active set $\mathcal{A}$ of M bricks parametrized by $\boldsymbol{\theta}$, we define the brick array $\mathbf{F}(\boldsymbol{\theta}) \in \mathbb{R}^{L\times m\times M}$ as the bricks evaluated at the datapoints $\mathbf{F}(\boldsymbol{\theta}) = \left[f_1^{j_1}(\mathcal{X};\boldsymbol{\vartheta}_1^{j_1}) \quad \dots \quad f_M^{j_M}(\mathcal{X};\boldsymbol{\vartheta}_M^{j_M})\right]$, where $f(\mathcal{X};\cdot)$ is the element-wise evaluation of the brick on the dataset $\mathcal{X}$. The residual of the model is thus defined as $r(\boldsymbol{a},\boldsymbol{\theta}) = \mathcal{Y} - \mathbf{F}(\boldsymbol{\theta})\boldsymbol{a}$.

Linear Fitting of Amplitudes A key feature of the LIESOLVER's structure (12) is the separability of parameters. For any fixed nonlinear configuration $\boldsymbol{\theta}$, the optimal amplitudes $\boldsymbol{a}^\star$ can be determined via linear regression. To ensure numerical stability, we employ Ridge regression with regularization parameter $\lambda > 0$:

$$\begin{aligned}\boldsymbol{a}^\star(\boldsymbol{\theta}) &= \arg\min_{\boldsymbol{a}}\left(\|\mathcal{Y} - \mathbf{F}(\boldsymbol{\theta})\boldsymbol{a}\|_2^2 + \lambda\,\|\boldsymbol{a}\|_2^2\right)\\ &= \left(\mathbf{F}(\boldsymbol{\theta})^\top\mathbf{F}(\boldsymbol{\theta}) + \lambda\, I\right)^{-1}\mathbf{F}(\boldsymbol{\theta})^\top\mathcal{Y}\ .\end{aligned} \tag{13}$$

Greedy Brick Initialization Initializing all M bricks simultaneously would lead to a stiff optimization landscape due to the heterogeneous nonlinearity of brick functions. Instead, we use a greedy selection strategy and build the active set $\mathcal{A}$ iteratively. At each step, we generate a candidate pool of random parameters $\{\boldsymbol{\vartheta}_p^j\}_{j=1\dots N,\, p=1\dots P}$. We select the candidate that maximizes the cosine similarity score with the current residual $r(\boldsymbol{a},\boldsymbol{\theta})$:

$$s_{j,p}(r) = \frac{\left|\langle r, f^j(\mathcal{X};\boldsymbol{\vartheta}_p^j)\rangle\right|}{\|r\|_2\ \left\|f^j(\mathcal{X};\boldsymbol{\vartheta}_p^j)\right\|_2}\ , \tag{14}$$

where $f^j(\mathcal{X};\boldsymbol{\vartheta}_p^j)$ is the evaluation of the candidate brick. Note that the dependency on $\boldsymbol{a}$ factors out in (14). The candidate with the highest score is added to $\mathcal{A}$ and $\boldsymbol{a}$ is updated via Ridge regression.

Nonlinear Refinement of Symmetry Parameters Greedy construction provides a robust initialization that eases finding a local minimum. To navigate the potentially stiff landscape, we employ nonlinear least squares (NLLS) optimization. The separability of the model structure (12) into linear and nonlinear parameters enables the use of the Variable Projection (VarPro) method (O'Leary & Rust, 2013). Instead of optimizing $(\boldsymbol{a},\boldsymbol{\theta})$ jointly, we treat the amplitudes as dependent variables $\boldsymbol{a}^\star(\boldsymbol{\theta})$ that implicitly solve the linear subproblem (13) for any given $\boldsymbol{\theta}$. This reduces the optimization objective to solely the nonlinear parameters:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}}\|\mathcal{Y} - \mathbf{F}(\boldsymbol{\theta})\,\boldsymbol{a}^\star(\boldsymbol{\theta})\|_2^2\ . \tag{15}$$

We solve (15) using the Trust Region Reflective algorithm (Branch et al., 1999). It handles parameter bounds explicitly, ensuring the validity of applied symmetry transformations. As a second-order method, it effectively resolves stiffness, particularly when supplied with Jacobians derived from our symbolic brick definitions. To ensure flexibility during training, we apply VarPro optimization both partially, to refine the most recently added bricks, and globally, to refine all parameters of the model.

Algorithm 1 Basic LieSolver

Require: IBC data $(\mathcal{X},\mathcal{Y})$; brick families $\mathcal{S}$; hyperparameters `mse_tol`, `max_bricks`, R
Ensure: bricks $\mathcal{A}$, amplitudes $\boldsymbol{a}$

```
A ← ∅, a ← [ ], r ← Y, mse ← ||r||_2^2 / L
while mse > mse_tol and |A| < max_bricks do
    for 1 ... R do
        (j*, p*) ← arg max_{j,p} s_{j,p}(r)                ▷ via (14)
        append f^{j*}(x; ϑ_{p*}^{j*}) to A
        a ← Ridge(F(θ), Y)                                 ▷ via (13)
        r ← Y − F(θ)a;   mse ← ||r||_2^2 / L
        if mse ≤ mse_tol or |A| = max_bricks then
            break
        end if
    end for
    (a, θ) ← VarPro(F(θ), Y)                               ▷ via (15), (13)
    r ← Y − F(θ)a;   mse ← ||r||_2^2 / L
end while
```

The LIESOLVER Algorithm We present the pseudocode for the basic implementation of LIESOLVER in Algorithm 1. The procedure alternates between a *structural learning phase*, consisting of R greedy additions, and a *global parameter refinement*, repeating this cycle until the convergence criteria are met.

4.3. The LIESOLVER Pipeline

The workflow (see Figure 1) of LIESOLVER begins with the construction of the set $\mathcal{S}$ of brick families. One needs to compute Lie symmetries T_ϑ (see details in Appendix B.2) and select start solutions f_{start} to compose bricks according to (11). The choice of brick families determines model expressivity. To guide this choice, we perform a *brick preselection*: we compose random Lie transformation sequences with start solutions, run LIESOLVER on generic IBVPs, and retain the families most frequently selected by the greedy al-

gorithm. The preselection of expressive bricks is performed once per PDE and reused across all IBVPs. Parameter bounds must also be defined to ensure the functions remain well-defined within the domain and avoid singularities. However, small symmetry parameters $|\vartheta|$ are always valid by locality of Lie groups.

We then sample points on the boundary parts Γ_k to construct $\mathcal{X}_k$. To ensure efficient coverage of the boundary components Γ_k, we employ Halton sampling (Halton, 1960) for training points. Standard uniform sampling is employed to generate IBC points for testing generalization and domain points for validating the solution against ground truth.

Finally, the LIESOLVER configuration requires selecting hyperparameters: the candidate pool size P, the refinement interval R, and the maximum number of function evaluations `nfev` for the NLLS optimization (15). These settings control the navigation in the loss landscape. While large P and `nfev` ensure the current active set is at a local minimum, this rigidity can impede the effectiveness of subsequent greedy additions. Thus, balanced settings are preferred, as they maintain the flexibility required for the model to navigate the landscape efficiently and achieve lower final errors. Upon convergence, LIESOLVER yields an interpretable symbolic expression for the solution.

4.4. Scope of LIESOLVER

The prevailing ML approach for IBVPs, the PINN framework, optimizes two often competing objectives: the PDE residual and the initial-boundary data fit. This means a lower total loss does not guarantee higher solution accuracy (Krishnapriyan et al., 2021). Furthermore, balancing these terms often requires complex re-weighting schemes or hard-constraint architectures (Lu et al., 2021b).

LIESOLVER circumvents this by enforcing the PDE as a hard constraint within the model structure, reducing the optimization to supervised regression on the initial-boundary data. For well-posed IBVPs, the ratio of domain error to boundary error is upper bounded. This guarantees *loss alignment*: minimizing the boundary residuals provably minimizes the global error. Consequently, the boundary loss serves as an intrinsic reliability metric that enables rigorous error estimation (see details in Appendix A.2).

Table 1 contrasts LIESOLVER with NNs, PINNs, and classical numerical methods, such as finite difference, finite volume, and finite element methods (FDM, FVM, FEM). Unlike classical methods, LIESOLVER is mesh-free, mitigates the curse of dimensionality, and allows for seamless fusion of sparse observational data. Compared to PINNs, our method yields a model with fewer parameters, expressed as weighted sums of analytical brick functions, offering symbolic interpretability.

The primary limitation of LIESOLVER is its restriction to linear homogeneous PDEs. While this class covers many fundamental equations in physics and engineering, extending the method to nonlinear systems remains future work. Additionally, the current pipeline requires the heuristic selection of brick families. While the computation of Lie symmetries can be automated (Hereman, 1996), automating the optimal brick composition is an open challenge.

5. Experiments

5.1. Experimental setup

Studied IBVPs To evaluate the performance and demonstrate the capabilities of LIESOLVER, we study various IBVPs based on the heat equation $f_t - f_{xx} = 0$ and the wave equation $f_{tt} - f_{xx} = 0$. To assess robustness, we select five initial condition (IC) profiles for each PDE, varying in regularity from smooth periodic functions to sharp, discontinuous steps (see Appendix C.3 for details). We evaluate the models on the domain $(x, t) \in (0, 1) \times (0, 0.1)$ for the heat equation and $(0, 1) \times (0, 1)$ for the wave equation. We set the boundary conditions (BCs) to constant values consistent with the ICs. For the wave equation, we additionally enforce zero initial velocity. We discretize these constraints using 1000–1500 collocation points per boundary.

Brick Solutions LIESOLVER requires a preselected set of parametrized brick solutions $f(x; \boldsymbol{\vartheta})$ as discussed in Section 4.3. For the heat equation, we use the start solutions $f_{\text{start},1}(x, t) = 1$ and $f_{\text{start},2}(x, t) = e^{-t} \sin x$, and construct three complementary brick families generated by the Lie symmetries (4)–(9):

- $f^1(x, t; \boldsymbol{\vartheta}^1) = T^4_{\vartheta_2} \cdot T^1_{\vartheta_1} \circ \left(e^{-t} \sin x\right)$: sine mode with variable phase and frequency.
- $f^2(x, t; \boldsymbol{\vartheta}^2) = T^1_{\vartheta_2} \cdot T^6_{\vartheta_1} \circ (1)$: diffusion-generated Gaussian pattern with variable centre and width.
- $f^3(x, t; \boldsymbol{\vartheta}^3) = T^1_{\vartheta_4} \cdot T^6_{\vartheta_3} \cdot T^4_{\vartheta_2} \cdot T^1_{\vartheta_1} \circ \left(e^{-t} \sin x\right)$: Gaussian pattern modulated by a sine wave with four variable parameters.

For the wave equation, even a subgroup of the full Lie symmetry group comprising spatial translation $T^1_\vartheta : f(x, t) \mapsto f(x - \vartheta, t)$ and spatiotemporal scaling $T^2_\vartheta : f(x, t) \mapsto f(e^\vartheta x, e^\vartheta t)$ yields highly expressive solutions. Furthermore, we enforce the zero initial velocity condition by requiring $\partial_t f|_{t=0} = 0$ in the start solutions, ensuring that every brick satisfies this constraint by construction:

- $f^1(x, t; \boldsymbol{\vartheta}^1) = T^2_{\vartheta_2} \cdot T^1_{\vartheta_1} \circ \left(\sin x \cos t\right)$: standing wave with variable phase and frequency.
- $f^2(x, t; \boldsymbol{\vartheta}^2) = T^1_{\vartheta_2} \cdot T^2_{\vartheta_1} \circ (e^{-(x-t)^2} + e^{-(x+t)^2})$: Gaussian pattern with variable centre and width. The condition of zero initial velocity leads to a counter-

propagating pair of travelling waves rather than a single pulse.

Training configuration We evaluate model performance using the mean squared error (MSE) on the test IBC data and the Relative L^2 error (L2RE) on the domain Ω. The L2RE is computed against analytical reference solutions f_{ref} derived via Fourier series expansion (see Appendix C.1). Since all target values are of order unity, the MSEs are directly comparable across instances. For the heat equation, we use an empirically determined configuration: $P = 1000$, $R = 5$, $\texttt{nfev} = 4$, $\lambda = 10^{-1}$, and $L = 3000$, with $\texttt{mse_tol} = 10^{-6}$. For the wave equation, $L = 5000$ and we apply partial parameter refinement every 5 brick additions and global every 20. We benchmark against a standard PINN trained with a hybrid Adam/L-BFGS optimizer to minimize a composite loss of PDE residuals and weighted IBC errors (see details in Appendix C.2).

5.2. Experimental Results

Table 2 summarizes the performance of LIESOLVER and PINNs for the considered IBC cases. Reported medians are computed over 5 independent seeds. For all ICs except the *Step* IC, LIESOLVER reliably attains IBC MSE values of order 10^{-6} using between 5 and 40 bricks. The *Step* IC requires more bricks due to its sharp jump. PINNs match LIESOLVER in terms of attained MSE only on the easiest cases, but fall short on the more challenging ICs.

LIESOLVER is about one to three orders of magnitude faster than PINNs across the considered cases. However, its runtime[7] increases sharply as the number of bricks grows. This growth is dominated by the global parameter refinement stage. More targeted refinement strategies could mitigate this bottleneck.

The preselected brick families strongly influence performance. Employing sine and Gaussian bricks, LIESOLVER naturally fits IC profiles with similar sine and Gaussian patterns and reaches the target MSE in less than a minute. The *Step* IC remains a challenging stress test and highlights model capability. In the following, we discuss two representative cases that illustrate LIESOLVER behaviour. The full results can be found in Appendix D.

Heat Equation with *Gauss* IC Figure 2a shows the evolution of metrics as a function of the number of added bricks. The greedy addition begins with a Gaussian brick f^2 that already captures most of the IC (MSE $\sim 10^{-3}$), followed by several oscillatory additions (f^1 and f^3) that enforce the Dirichlet BCs. The first global refinement stage after 5 bricks yields a one-order-of-magnitude drop in MSE; the target tolerance ($< 10^{-6}$) is met after 15 bricks. Details on performance metrics and runtime are provided in Table 2. Moreover, we observe that the ratio between the domain MSE and the IBC test MSE remains bounded throughout training, consistent with the theoretical expectations in Appendix A.2. Thus, the IBC test MSE serves as a reliable measure for the domain MSE.

[7] All experiments were run on an Intel Core Ultra 7 165H CPU with 11 cores and 16 GiB RAM.

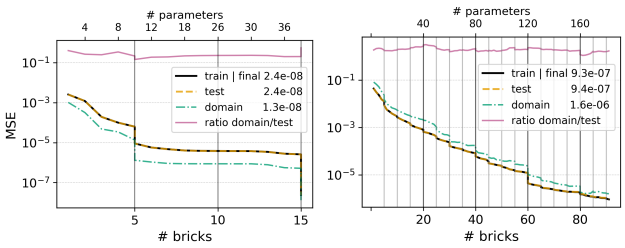
Table 2. Comparison of LIESOLVER and PINN performance. 'Params' denotes the network architecture for PINNs and the number of nonlinear parameters for LIESOLVER. Reported values correspond to the median across 5 independent seeds. Standard deviations are omitted for clarity, as the performance gap is stable across all trials.

Case	Method	MSE (IBC)	L2RE (Domain)	Time[7] [s]	Params
Heat *Poly*	LIESOLVER	$\mathbf{8.2 \cdot 10^{-7}}$	$\mathbf{4.4 \cdot 10^{-4}}$	**34**	86
	PINN	$9.0 \cdot 10^{-7}$	$4.8 \cdot 10^{-4}$	137	4×50
Heat *Gauss*	LIESOLVER	$\mathbf{2.4 \cdot 10^{-8}}$	$\mathbf{2.0 \cdot 10^{-4}}$	**11**	40
	PINN	$3.2 \cdot 10^{-7}$	$4.9 \cdot 10^{-4}$	123	4×50
Heat *Sine*	LIESOLVER	$\mathbf{2.4 \cdot 10^{-8}}$	$\mathbf{4.6 \cdot 10^{-4}}$	**4**	12
	PINN	$7.8 \cdot 10^{-7}$	$5.0 \cdot 10^{-3}$	183	4×50
Heat *Sine Mix*	LIESOLVER	$\mathbf{2.0 \cdot 10^{-8}}$	$\mathbf{3.5 \cdot 10^{-4}}$	**10**	38
	PINN	$4.6 \cdot 10^{-6}$	$1.9 \cdot 10^{-2}$	1716	6×100
Heat *Step*	LIESOLVER	$\mathbf{9.1 \cdot 10^{-7}}$	$\mathbf{1.3 \cdot 10^{-3}}$	**310**	230
	PINN	$1.8 \cdot 10^{-3}$	$2.2 \cdot 10^{-1}$	440	6×100
Wave *Gauss*	LIESOLVER	$\mathbf{4.9 \cdot 10^{-8}}$	$\mathbf{1.1 \cdot 10^{-3}}$	**8**	30
	PINN	$1.5 \cdot 10^{-7}$	$1.6 \cdot 10^{-3}$	1119	4×50
Wave *Gauss Mix*	LIESOLVER	$\mathbf{7.8 \cdot 10^{-7}}$	$\mathbf{5.0 \cdot 10^{-3}}$	**41**	82
	PINN	$2.7 \cdot 10^{-6}$	$8.2 \cdot 10^{-3}$	2456	4×50
Wave *Sine*	LIESOLVER	$\mathbf{1.1 \cdot 10^{-7}}$	$\mathbf{8.1 \cdot 10^{-4}}$	**5**	18
	PINN	$2.0 \cdot 10^{-7}$	$8.5 \cdot 10^{-4}$	1172	4×50
Wave *Sine Mix*	LIESOLVER	$\mathbf{4.6 \cdot 10^{-7}}$	$\mathbf{1.7 \cdot 10^{-3}}$	**4**	14
	PINN	$8.2 \cdot 10^{-5}$	$1.8 \cdot 10^{-2}$	11008	6×100
Wave *Step*	LIESOLVER	$\mathbf{9.4 \cdot 10^{-7}}$	$\mathbf{4.0 \cdot 10^{-3}}$	**217**	182
	PINN	$7.2 \cdot 10^{-5}$	$4.5 \cdot 10^{-2}$	909	4×50

Figures 3 and 4 compare the model prediction with the ground truth on the IBCs and in the domain, respectively. The IBC plots include a decomposition into the weighted bricks, revealing dominant Gaussian-like contributors for the IC profile and additional bricks that fit the BCs. For reference, a vanilla PINN also fits this smooth case to an IBC MSE $< 10^{-6}$, albeit at a longer runtime, as indicated in Table 2.

Wave Equation with *Step* IC The *Step* IC presents the most challenging case due to the jump in the profile. Figure 2b highlights two optimization regimes: (i) a gradual decrease in MSE where greedy initialization with refinement brings limited improvement, and (ii) rapid progress once sufficient expressivity is reached. Notably, a global refinement at 60 bricks yields a pronounced drop in error. Furthermore, we again observe that the domain error tracks the IBC error throughout the training.

Figure 5 shows that the solution is primarily composed of standing-wave modes with localized Gaussian correc-



(a) Heat PDE with *Gauss* IC *(b)* Wave PDE with *Step* IC

Figure 2. Fit progress of LIESOLVER: metrics are shown as functions of the number of added bricks. The black vertical lines correspond to the global refinement steps, grey lines correspond to partial refinements of the last 5 added bricks.

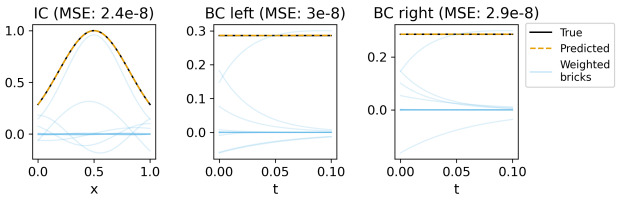


Figure 3. IBC fits for the heat equation with *Gauss* IC, including the decomposition into weighted brick contributions. The transparent blue curves depict the individual contributions.

tions around the discontinuity. During optimization, the greedy stage first selects sine bricks to fit the dominant high-amplitude oscillatory component. As the remaining residual becomes localized and irregular, the algorithm increasingly adds Gaussian bricks to correct for edges and boundaries. This localization is evident in Figure 6, where the error density is concentrated along the propagating wavefronts ($x \pm t$). PINNs do not resolve the sharp feature sufficiently well and achieve only an L2RE of $4.5 \cdot 10^{-2}$ in the domain (see Appendix D.2). Overall, the results demonstrate that LIESOLVER consistently achieves higher accuracy and significantly faster convergence than PINNs, particularly in cases involving complex or sharp initial conditions.

6. Conclusion

Using Lie symmetries, we generate parametrized functions that exactly solve the corresponding PDE. By incorporating

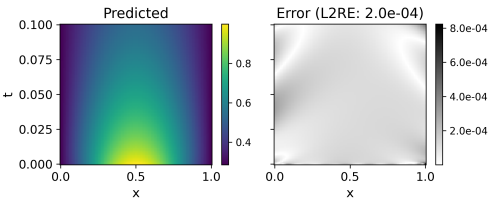


Figure 4. 2D domain fields (prediction and error) for the heat equation with *Gauss* IC.

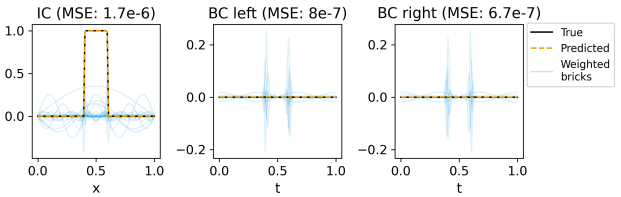


Figure 5. IBC fits for the wave equation with *Step* IC, including the decomposition into weighted brick contributions. The transparent blue curves depict the individual contributions.

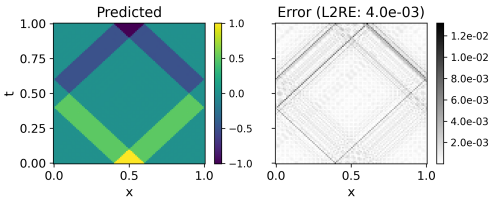


Figure 6. 2D domain fields (prediction and error) for the wave equation with *Step* IC.

the underlying symmetry structures directly into the model, solving IBVPs reduces to optimizing these parameters to satisfy the initial-boundary conditions (IBCs). Focusing on the common case of linear homogeneous PDEs, we introduce LIESOLVER, which represents the solution as a linear combination of Lie-symmetry-generated brick solutions whose linear amplitudes and nonlinear symmetry parameters are learned from the IBCs. LIESOLVER constructs this combination incrementally, alternating between greedily initializing bricks best aligned with the residual and refining parameters.

By incorporating the PDE into the model's structure, LIESOLVER offers several advantages: the loss function effectively serves as a faithful indicator of predictive accuracy, so that — unlike in PINNs — a decrease in the objective on the boundary directly corresponds to an improvement in accuracy on the domain. In addition, this property enables rigorous error bounds for well-posed IBVPs, in contrast to PINNs, where such guarantees are generally missing and are an active field of research. Moreover, these tailor-made models use comparatively few parameters, which results in significant computational speedups compared to PINNs without sacrificing expressivity. Furthermore, LIESOLVER expresses its prediction as a sparse, analytical expression, thereby enhancing interpretability.

Across all benchmarks on various IBVPs, LIESOLVER consistently outperformed PINNs in both accuracy and computational efficiency, running one to three orders of magnitude faster. For smooth ICs, LIESOLVER achieved an IBC MSE below 10^{-6} with fewer than 100 nonlinear parameters. For the harder step-like ICs, more parameters were required, yet LIESOLVER still obtained a domain L2RE on the order of

10^{-3}, whereas PINNs plateaued between 10^{-2} and 10^{-1}. In summary, LIESOLVER delivers substantially higher accuracy at a fraction of the computational cost, establishing it as a highly effective alternative to PINNs for solving IBVPs.

Limitations Despite the notable advantages of using Lie symmetries, it is important to acknowledge the limitations of our approach. Currently, LIESOLVER applies only to linear homogeneous PDEs. Although this encompasses a large class of important problems, we aim to generalize our framework in future work. Another requirement is that for every PDE, appropriate bricks — composed from Lie symmetries — and at least one start solution must be selected. We address this through a systematic preselection procedure (Section 4.3), which identifies expressive bricks. Lie symmetries themselves can be determined algorithmically (Hereman, 1996) or found in the literature for standard problems (Olver, 1993). However, as is the case in conventional NNs, performance highly depends on various design choices. Nonetheless, in our approach, the loss provides a rigorous criterion for assessing whether a particular choice of brick solution is effective or needs to be extended, providing useful criteria for these design choices.It is also important to note that we have tested LIESOLVER only on a limited number of problems. Therefore, more extensive studies will be necessary in future work.

These findings suggest several promising directions: broader empirical evaluation, complete automation from symmetry derivation to brick preselection, and extension to general PDEs. The extension to inhomogeneous linear PDEs is straightforward: the difference of any two solutions of the inhomogeneous PDE is a solution of the homogeneous PDE. Hence, by adding an arbitrary particular solution to (12), one can represent the full inhomogeneous solution space. For nonlinear PDEs, Lie symmetry theory still applies, but superposition no longer holds in general. Here, genetic algorithms and evolutionary approaches appear to be promising directions, since the lack of superposition removes the linear factorization our approach currently exploits. By pursuing these steps, we aim to enhance the capabilities and applicability of LIESOLVER and provide an interpretable, reliable, and efficient IBVP solver for general PDEs.

Code availability

The Python implementation is available at `https://github.com/oduwancheekee/liesolver`.

Impact Statement

This work aims to advance the field of scientific machine learning by proposing an alternative method for solving IBVPs. We do not foresee specific societal consequences beyond those already associated with existing PDE solvers.

References

Akhound-Sadegh, T., Perreault-Levasseur, L., Brandstetter, J., Welling, M., and Ravanbakhsh, S. Lie Point Symmetry and Physics-Informed Networks. *Advances in Neural Information Processing Systems*, 36:42468–42481, 2023. arXiv: 2311.04293.

Bluman, G. W. and Kumei, S. *Symmetries and differential equations*. Applied Mathematical Sciences. Springer, 1989.

Bluman, G. W., Cheviakov, A. F., and Anco, S. C. *Applications of symmetry methods to partial differential equations*. Applied mathematical sciences. Springer, 2010.

Boyce, W. E. *Elementary differential equations and boundary value problems, eleventh edition loose-leaf print companion*. John Wiley & Sons, 11 edition, 2016.

Branch, M. A., Coleman, T. F., and Li, Y. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

Brandstetter, J., Welling, M., and Worrall, D. E. Lie Point Symmetry Data Augmentation for Neural PDE Solvers. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 2241–2256. PMLR, 2022.

Butcher, J., Carminati, J., and Vu, K. A comparative study of some computer algebra packages which determine the lie point symmetries of differential equations. *Computer Physics Communications*, 155(2):92–114, 2003.

Champagne, B., Hereman, W., and Winternitz, P. The computer calculation of lie point symmetries of large systems of differential equations. *Computer Physics Communications*, 66(2):319–340, 1991.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pp. 2990–2999. PMLR, 2016.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. Lagrangian neural networks. In *International Conference on Learning Representations*, 2020.

Duistermaat, J. J. and Kolk, J. A. C. *Lie groups*. Springer, 1999.

Eiras, F., Bibi, A., Bunel, R., Dvijotham, K. D., Torr, P., and Kumar, M. P. Efficient error certification for physics-informed neural networks, 2024. arXiv: 2305.10157.

Evans, L. C. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, 2nd edition, 2010.

Fairweather, G. and Karageorghis, A. The method of fundamental solutions for elliptic boundary value problems. *Advances in computational mathematics*, 9(1):69–95, 1998.

Greydanus, S., Dzamba, M., and Yosinski, D. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Halton, J. H. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2(1):84–90, 1960.

Hereman, W. Symbolic software for lie symmetry analysis. In Ibragimov, N. (ed.), *CRC Handbook of Lie Group Analysis of Differential Equations*, volume III, pp. 367–413. CRC Press, 1996.

Hillebrecht, B. and Unger, B. Prediction error certification for pinns: Theory, computation, and application to stokes flow, 2025. arXiv: 2508.07994.

Hiptmair, R., Moiola, A., and Perugia, I. A survey of trefftz methods for the helmholtz equation. In *Building bridges: connections and challenges in modern approaches to numerical partial differential equations*, pp. 237–279. Springer, 2016.

Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.

John, F. *Partial Differential Equations*. Applied mathematical sciences. Springer US, 1978. ISBN 978-1-4684-0061-8.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kołodziej, J. A. and Grabski, J. K. Many names of the Trefftz method. *Engineering Analysis with Boundary Elements*, 96:169–178, 2018.

Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks, 2021. arXiv: 2109.01050.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. Artificial Neural Networks for Solving Ordinary and Partial Differential Equations. *IEEE Transactions on Neural Networks*, 9(5): 987–1000, 1998. arXiv: physics/9705023.

Lau, G. K. R., Hemachandra, A., Ng, S.-K., and Low, B. K. H. PINNACLE: PINN Adaptive ColLocation and Experimental points selection. In *The Twelfth International Conference on Learning Representations*, 2023.

LeVeque, R. J. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Society for Industrial and Applied Mathematics, 2007. ISBN 978-0-89871-629-0.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljacic, M., Hou, T. Y., and Tegmark, M. KAN: Kolmogorov-arnold networks. *ICLR*, 2025. arXiv: 2404.19756.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021a.

Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., and Johnson, S. G. Physics-informed neural networks with hard constraints for inverse design, 2021b. arXiv: 2102.04626.

Meng, X., Li, Z., and Karniadakis, G. E. Ppinns: Physics-informed neural networks with prior. *Journal of Computational Physics*, 401:109020, 2020.

Mialon, G., Garrido, Q., Lawrence, H., Rehman, D., LeCun, Y., and Kiani, B. Self-Supervised Learning with Lie Symmetries for Partial Differential Equations. *Advances in Neural Information Processing Systems*, 36:28973–29004, 2023.

Naujoks, J. R., Krasowski, A., Weckbecker, M., Yolcu, G. U., Wiegand, T., Lapuschkin, S., Samek, W., and Klausen, R. P. Leveraging Influence Functions for Resampling Data in Physics-Informed Neural Networks, 2025. arXiv: 2506.16443.

O'Leary, D. P. and Rust, B. W. Variable projection for nonlinear least squares problems. *Computational Optimization and Applications*, 54(3):579–593, 2013.

Oliveri, F. Lie symmetries of differential equations: Classical results and recent contributions. *Symmetry*, 2(2): 658–706, 2010.

Olver, P. J. Symmetry and explicit solutions of partial differential equations. *Applied Numerical Mathematics*, 10 (3):307–324, 1992.

Olver, P. J. *Applications of Lie Groups to Differential Equations*. Springer-Verlag, 2nd edition, 1993.

Olver, P. J. *Introduction to partial differential equations*. Undergraduate texts in mathematics. Springer International Publishing, 1 edition, 2013.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Rathore, P., Lei, W., Frangella, Z., Lu, L., and Udell, M. Challenges in training PINNs: A loss landscape perspective. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42159–42191. PMLR, 2024.

Rauch, J. *Partial Differential Equations*. Graduate Texts in Mathematics. Springer, 1991. ISBN 978-0-387-97472-9.

Rudin, C. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead, 2019. arXiv: 1811.10154.

Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., and Müller, K.-R. Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. *Proceedings of the IEEE*, 109(3):247–278, 2021. arXiv: 2003.07631.

Steinberg, S. and de Melo Marinho Junior, R. Practical guide to the symbolic computation of symmetries of differential equations, 2014. arXiv: 1409.8364.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Trefftz, E. Ein Gegenstück zum Ritzschen Verfahren. *Proc. 2nd Int. Congress for Applied Mechanics, Zürich*, 1926.

Tsoulos, I. G. and Lagaris, I. E. Solving differential equations with genetic programming. *Genetic Programming and Evolvable Machines*, 7(1):33–54, 2006.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Walczak, M., Pfrommer, J., Pick, A., Ramamurthy, R., Garcke, J., Bauckhage, C., and Schuecker, J. Informed machine learning - a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

Wang, S., Teng, Y., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Wang, S., Sankaran, S., Wang, H., and Perdikaris, P. An expert's guide to training physics-informed neural networks, 2023. arXiv: 2308.08468.

Wang, S., Bhartari, A. K., Li, B., and Perdikaris, P. Gradient alignment in physics-informed neural networks: A second-order optimization perspective, 2025. arXiv: 2502.00604.

Wigner, E. P. *Symmetries and reflections: scientific essays*. Indiana University Press, 1967.

Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

Yang, J., Dehmamy, N., Walters, R., and Yu, R. Latent space symmetry discovery. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pp. 56047–56070. JMLR.org, 2024a.

Yang, J., Rao, W., Dehmamy, N., Walters, R., and Yu, R. Symmetry-Informed Governing Equation Discovery. *Advances in Neural Information Processing Systems*, 37: 65297–65327, 2024b.

Zhang, Z.-Y., Cai, S.-J., and Zhang, H. A symmetry group based supervised learning method for solving partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 414:116181, 2023a.

Zhang, Z.-Y., Zhang, H., Zhang, L.-S., and Guo, L.-L. Enforcing continuous symmetries in physics-informed neural network for solving forward and inverse problems of partial differential equations. *Journal of Computational Physics*, 492:124145, 2023b.

A. Extended Discussion of Initial-Boundary Value Problems

Initial-boundary value problems (IBVPs) are essential for modelling various phenomena in physics and many other quantitative sciences. Since they are the main object of study in this article, we provide some useful background information. In contrast to the summary in Section 3.1, we will introduce partial differential equations (Appendix A.1) and IBVPs (Appendix A.2) in a way, which is most convenient to discuss their symmetries in Appendix B. We split this overview in the two components of IBVPs: partial differential equations and initial-boundary conditions (IBCs).

A.1. Partial Differential Equation (PDEs)

In a nutshell, PDEs are just equations containing derivatives. Thus, we can consider them for example as differential operators acting on functions, as done in Section 3.1. However, when studying symmetries of PDEs, it is often convenient to think of their solutions as parametrized surfaces, instead of functions, which enables a geometric treatment. In the following, we develop this idea rigorously and introduce the key concepts. This summary is oriented towards the description in (Olver, 1993). For a more general perspective on PDEs we recommend (Rauch, 1991; John, 1978; Evans, 2010; Olver, 2013).

Consider n *independent variables* $x = (x_1, \dots, x_n) \in X = \mathbb{R}^n$ and m *dependent variables* $u = (u^1, \dots, u^m) \in \mathcal{U} = \mathbb{R}^m$. This terminology reflects that we will later treat solutions as maps from X to $\mathcal{U}$. In addition to the dependent and independent variables, the PDE also contains derivatives. Therefore, we denote by $d \in \mathbb{N}$ the *order* of the considered PDE, i.e., the order of its highest derivative. For n independent variables, the number of distinct partial derivatives up to order d is given[8] by $\binom{n+d}{d}$. We *prolong* the space $\mathcal{U}$ to include all partial derivatives up to order d, which we denote by $\mathcal{U}^{(d)} := \mathbb{R}^{m\binom{n+d}{d}}$. By these definitions, a *system of p PDEs of order d* is given as a set of equations

$$\mathcal{D}_j\big(x, u^{(d)}\big) = 0, \qquad j = 1, \dots, p \tag{16}$$

where we assume $\mathcal{D} : X \times \mathcal{U}^{(d)} \to \mathbb{R}^p$ to be smooth. Accordingly, we describe PDEs as functions of the independent variables and the prolonged dependent variables.

In order to connect the placeholder variable $u^{(d)}$ to the actual derivatives, we introduce the *prolongation* of functions. For a sufficiently smooth function $f : \Omega \to \mathcal{U}$ with $\Omega \subseteq X$ open, its d-th prolongation $\mathrm{pr}^{(d)} f : \Omega \to \mathcal{U}^{(d)}$ is defined as the tuple

$$\mathrm{pr}^{(d)} f(x) := \Big(\partial_J f(x)\Big)_{J \in [n]^d} \in \mathcal{U}^{(d)} \ , \tag{17}$$

where $[n]^d := \big\{\sigma \subseteq \{1, \dots, n\} : |\sigma| \leq d\big\}$ contains all subsets of $[n]$ with at most d elements (including the empty set), and we write for a set $J = \{j_1, \dots, j_k\}$

$$\partial_j := \frac{\partial}{\partial x_j}, \qquad \partial_{\{j_1,\dots,j_k\}} := \partial_{j_1} \cdots \partial_{j_k} \quad \text{and} \quad \partial_\emptyset f := f \tag{18}$$

for brevity. Thus, the prolongation explicitly lists all derivatives up to order d. For instance, for a function $f : \mathbb{R}^2 \to \mathbb{R}$ we get $\mathrm{pr}^{(2)} f(x, y) = \big(f, f_x, f_y, f_{xx}, f_{xy}, f_{yy}\big) \in \mathcal{U}^{(2)} \cong \mathbb{R}^6$.

A sufficiently smooth function $f : \Omega \to \mathcal{U}$, with $\Omega \subseteq X$ open, is called a *solution of the PDE system $\mathcal{D}$* if

$$\mathcal{D}(x, \mathrm{pr}^{(d)} f(x)) = 0 \qquad \forall x \in \Omega \ . \tag{19}$$

It is important to note, that there is no general theory to find solutions of PDEs. However, for the further treatment, it is useful to classify PDEs with respect to their properties. We call a PDE (system) $\mathcal{D}$ *linear* if we can write it in the form

$$\mathcal{D}(x, \mathrm{pr}^{(d)} f(x)) = h(x) + \sum_{J \in [n]^d} a_J(x) \partial_J f(x) \tag{20}$$

for some functions $h, a_J : X \to \mathbb{R}^p$. If in addition $h(x) \equiv 0$, we call the PDE (system) *linear homogeneous*. For any linear homogeneous PDEs, we have that if f_1 and f_2 are two solutions of the PDE, then so is $\alpha f_1 + \beta f_2$ for any $\alpha, \beta \in \mathbb{R}$. In particular, it holds that $f(x) \equiv 0$ is always a solution for any linear homogeneous PDE. If a linear homogeneous PDE does not contain a term depending on u, it additionally is true that every constant function solves this PDE.

[8] We always assume sufficiently smooth functions, such that derivatives will commute. The number of partial derivatives follows from the combinatorial identity $\sum_{i=0}^{d} \binom{n+i-1}{i} = \binom{n+d}{d}$ for $d \geq 0$ and $n > 0$, which can be easily shown by induction.

A.2. Initial-Boundary Conditions and Well-Posed IBVPs

Since PDEs usually admit infinitely many solutions, we impose additional constraints in the form of *initial-boundary conditions* (IBC) to ensure uniqueness. To simplify the notation, we will not explicitly distinguish between time and spatial coordinates. Hence, initial and boundary conditions get the same form. In accordance with Section 3.1, let $\Omega \subset X$ be a bounded domain, where its boundary $\partial\Omega$ contains smooth, connected components $\Gamma_1, \dots, \Gamma_r \subseteq \partial\Omega$. Analogously to PDEs, we can represent IBCs either by operators or by equations. For a consistent notation with Appendix A.1, which is also more oriented towards the way a computer handles these conditions, we describe them also by equations. Thus, for each Γ_k, we introduce a condition of the form

$$\mathcal{B}_k(x, u, u_{\boldsymbol{n}}) = 0 \quad \text{for all } x \in \Gamma_k \tag{21}$$

with $\mathcal{B}_k : \Gamma_k \times \mathcal{U} \times \mathcal{U} \to \mathbb{R}$. The symbol $u_{\boldsymbol{n}}$ is a placeholder, which will later be associated with the normal derivative. A PDE system $\mathcal{D}$ together with associated IBCs $\mathcal{B}_1, \dots, \mathcal{B}_r$ defines an *initial-boundary value problem* (IBVP). A sufficiently smooth function $f : \overline{\Omega} \to \mathcal{U}$ is called a *solution of the IBVP* if

$$\mathcal{D}(x, \mathrm{pr}^{(d)} f(x)) = 0 \qquad \forall x \in \Omega \tag{22}$$

$$\mathcal{B}_k\Big(x, f(x), \frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{n}}(x)\Big) = 0 \qquad \forall x \in \Gamma_k,\ k = 1, \dots, r \ , \tag{23}$$

where $\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{n}}(x) = \nabla f(x) \cdot \hat{n}(x)$ is the normal outward derivative on Γ_k.

In this work, we always focus on *well-posed* IBVPs, meaning that

(i) the IBVP has a solution,

(ii) the solution is unique, and

(iii) the solution depends continuously on the prescribed initial and boundary data.

Problems not satisfying (i) and (ii) are not expected to appear in the modelling of a concrete physical phenomenon. However, there are some nonlinear problems, which are not satisfying (iii), often referred to as chaotic systems. Due to their numeric instability, they are inaccessible by classical numerical methods. The precise definition of these three properties depends on the specific problem being considered, and there is no commonly agreed definition, as noted in (John, 1978).

In order to clarify the precise meaning of (i) and (ii), we typically specify the space of possible solutions to smooth functions and may add further restrictions, e.g., certain assumptions on the growth at infinity, in order to specify uniqueness. When formulating in metric spaces, the third condition (iii) means that the map from IBCs to solutions is (locally) Lipschitz continuous. To illustrate the last condition in practice, we give the following example.

Example A.1 [Heat Equation and Reverse Heat Equation]: Consider the so-called reverse heat equation in the domain $x \in [0, 2\pi)$, $t > 0$ with vanishing boundary condition and two different initial conditions

$$u_t + u_{xx} = 0,\ u(x,0) = 0,\ u(0,t) = u(2\pi,t) = 0 \tag{24}$$

$$v_t + v_{xx} = 0,\ v(x,0) = \epsilon \sin(x),\ v(0,t) = v(2\pi,t) = 0 \ , \tag{25}$$

with $\epsilon > 0$. It is not hard to verify the two solutions $u(x,t) \equiv 0$ and $v(x,t) = \epsilon \sin(x) e^t$. These two problems only differ in the initial condition, having the distance $\|u(0) - v(0)\|_\infty = \epsilon$, with the norm $\|f(t)\|_\infty := \max_{x\in[0,2\pi)} |f(x,t)|$. Hence, the initial conditions are arbitrarily close for small ϵ. However, the distance between solutions is given by $\|u(t) - v(t)\|_\infty = \epsilon\, e^t$, which is not bounded on the domain. Hence, the solution will not depend continuously on the initial data, and this problem is ill-posed.

In comparison, consider a slightly modified version of the problem, which is the usual heat equation

$$u_t - u_{xx} = 0,\ u(x,0) = 0,\ u(0,t) = u(2\pi,t) = 0 \tag{26}$$

$$v_t - v_{xx} = 0,\ v(x,0) = \epsilon \sin(x),\ v(0,t) = v(2\pi,t) = 0 \ , \tag{27}$$

which results in the solutions $u(x,t) \equiv 0$ and $v(x,t) = \epsilon \sin(x) e^{-t}$. In that case, the distance between the solutions is given by $\|u(t) - v(t)\|_\infty = \epsilon\, e^{-t} \leq \epsilon$. Hence, the map from initial data to solutions is (locally) Lipschitz continuous

$$\|u(t) - v(t)\|_\infty \leq C\, \|u(0) - v(0)\|_\infty \tag{28}$$

with Lipschitz constant $C = 1$. Hence, we confirmed the well-posedness of the heat equation with Dirichlet conditions. In other words, the ratio $\|u(t) - v(t)\|_\infty / \|u(0) - v(0)\|_\infty \leq C$ is bounded. $\triangle$

The precise form of (28) depends naturally on the formulation of the IBVP and gets more involved, when increasing the number of conditions and variables. However, we always have that the ratio between changes in domain values and changes in initial-boundary values is bounded from above. This particular property of well-posed IBVPs, leads to a rigorous error estimation of LIESOLVER: If we know the deviation on the boundary, we can directly infer the (maximal) error in the domain.

B. Extended Discussion of Lie Symmetries

We expand our summary from Section 3.2 by presenting additional foundational concepts, followed by a detailed example illustrating the systematic derivation of Lie symmetry transformations for a given PDE. For a comprehensive exploration of these subjects, we refer to (Olver, 1993) as an extensive standard source, (Olver, 1992; Bluman & Kumei, 1989; Bluman et al., 2010) for alternative presentations, (Duistermaat & Kolk, 1999) for a focus on Lie groups, and (Oliveri, 2010) for an overview, including a historical perspective.

B.1. Basics on Lie Symmetries and Lie Groups

PDEs can be invariant under certain symmetries, and so their set of solutions has to respect the same symmetries. Let us illustrate this behaviour on the simple differential equation $u_{xx} = 0$. Obviously, if $u = f(x)$ is a solution, then so is $u = f(x) + \vartheta$, as well as $u = e^\vartheta f(x)$ for any $\vartheta \in \mathbb{R}$. Hence, the underlying PDE comes with a translation and scaling invariance

$$T^1_\vartheta : f(x) \mapsto f(x) + \vartheta \quad \text{and} \quad T^2_\vartheta : f(x) \mapsto e^\vartheta f(x) \ . \tag{29}$$

Moreover, in this example, these symmetry transformations will generate a group structure, i.e., we have a neutral element $T^i_0 = e$, inversions $\left(T^i_\vartheta\right)^{-1} = T^i_{-\vartheta}$ and $T^i_{\vartheta_1} \circ T^i_{\vartheta_2} = T^i_{\vartheta_1+\vartheta_2}$. Thus, we obtain a two-dimensional group, which is spanned by two one-parameter groups.

The same structure persists for general PDEs. The rigorous study of these symmetry groups naturally leads to the framework of *Lie groups* (Duistermaat & Kolk, 1999), which provide the mathematical structure for continuous transformation groups. A Lie group is a smooth, finite-dimensional manifold endowed with a group structure[9] such that both the multiplication and inversion maps are smooth.

Symmetry transformations can be described as actions of the Lie groups on manifolds. For a Lie group G and a smooth manifold M, this action $\circ : G \times M \to M$ is required to be compatible with the group axioms, i.e., we demand $g_1 \circ (g_2 \circ x) = (g_1 \cdot g_2) \circ x$ and $e \circ x = x$ for any $g_1, g_2 \in G$ and all $x \in M$.

Depending on the group G and the manifold M, not all of these actions may be defined, which is why we allow restricting these actions to a subset $\mathscr{U} \subseteq G \times M$ containing the identity $\{e\} \times M \subseteq \mathscr{U}$. We call this restriction a *local group of transformations*. Note that this restriction to group elements around the identity does not pose any serious limitations, since every element of the full Lie group G can be written as a finite product of elements which are close to the identity (compare Olver, 1993, thm. 1.22 and prop. 1.24).

In order to connect a PDE solution to a manifold, let $f : \Omega \to \mathcal{U}$ be a sufficiently smooth function with $\Omega \subseteq X$ open. Its *graph* is defined as

$$\Gamma_f := \{(x, f(x)) \mid x \in \Omega\} \subseteq \Omega \times \mathcal{U} \ , \tag{30}$$

a smooth submanifold of $X \times \mathcal{U}$. If G is a (local) group of transformations acting on $X \times \mathcal{U}$, then for $g \in G$ we define the transformed graph as

$$g \circ \Gamma_f := \{(\tilde{x}, \tilde{u}) = g \circ (x, u) \mid (x, u) \in \Gamma_f\} \ . \tag{31}$$

In the case where we can write the transformed graph $g \circ \Gamma_f = \Gamma_{\tilde{f}}$ as a graph of another function $\tilde{f}$, we simply write $\tilde{f} = g \circ f$. Note that for elements g close to the identity, we always find a (single-valued) function $\tilde{f}$ with that property.

[9] A group $(G, \cdot)$ is a set G equipped with a binary operation $\cdot : G \times G \to G$ that is associative, has a unique identity element $e \in G$, and assigns to each $g \in G$ a unique inverse g^{-1}.

A *symmetry group of a PDE* $\mathcal{D}$ is a local group of transformations G acting on $M \subseteq X \times \mathcal{U}^{(d)}$, such that whenever $u = f(x)$ is a solution of $\mathcal{D}$, and $g \circ f$ is defined for $g \in G$, then $u = g \circ f(x)$ will be a solution of $\mathcal{D}$ as well. There are several constructive methods to determine the full symmetry group of a PDE, where we will discuss one option in the following section.

B.2. Computation of Lie Symmetries

We will close this short summary on Lie symmetries, by providing a brief overview of how to compute the symmetry group of a system. To keep the description as short as possible, we will omit many details and focus on a practical guide for calculating the symmetry group. For further details, we recommend (Olver, 1993, ch. 2), which the following outline is oriented towards. Before stating the so-called infinitesimal criterion, which is the key concept for the determination of symmetry groups, we have to introduce some further definitions.

Every Lie group G is related to a so-called *Lie algebra* $\mathfrak{g}$, which is defined to be the tangent space of G at its identity element (for further details see e.g. Duistermaat & Kolk, 1999). Due to their vector space structure, it is often much more convenient to work with the Lie algebra $\mathfrak{g}$, instead of the Lie group G. By Lie's third theorem (Olver, 1993, thm. 1.54) there is a one to one correspondence between finite-dimensional Lie algebras $\mathfrak{g}$ and connected Lie groups G. Thus, we lose no generality, when considering Lie algebras. Elements of the Lie algebras are also called the *infinitesimal generators* of a Lie group.

A system of PDEs $\mathcal{D}(x, u^{(d)})$ is called to be of *maximal rank*, if its Jacobian $J_{\mathcal{D}}(x, u^{(d)})$ with respect to all variables has the maximal rank p, whenever $\mathcal{D}(x, u^{(d)}) = 0$. E.g., $\mathcal{D}(x, u^{(2)}) = u_t - u_{xx}$ has Jacobian $J_{\mathcal{D}}(x, t; u, u_x, u_t, u_{xx}, u_{xt}, u_{tt}) = (0, 0; 0, 0, 1, -1, 0, 0)$, which is of maximal rank, whereas $\mathcal{D}(x, u^{(2)}) = (u_t - u_{xx})^2$ is not of maximal rank.

Theorem B.1 (Infinitesimal criterion (for a proof see Olver, 1993, thm. 2.31))**.** *Let $\mathcal{D}$ be a system of PDEs with maximal rank defined over $M \subseteq X \times U$ and let G be a local group of transformations acting on M. If*

$$\mathrm{pr}^{(d)}\, \mathbf{v}[\mathcal{D}(x, u^{(d)})] = 0, \quad \textit{whenever} \;\; \mathcal{D}(x, u^{(d)}) = 0 \tag{32}$$

holds for any infinitesimal generator $\mathbf{v}$ of G, then G is a symmetry group of $\mathcal{D}$.

In fact, this theorem gives us a computable criterion to determine the infinitesimal generators $\mathbf{v}$ of the symmetry group. To evaluate (32), we also need to prolong a vector fields, which can be computed for a general vector field

$$\mathbf{v} = \sum_{i=1}^{n} \xi^i(x, u) \frac{\partial}{\partial x_i} + \sum_{k=1}^{m} \phi_k(x, u) \frac{\partial}{\partial u^k} \tag{33}$$

by (Olver, 1993, thm. 2.36 and prop. 2.35)

$$\mathrm{pr}^{(d)}\, \mathbf{v} = \mathbf{v} + \sum_{k=1}^{m} \sum_{J \in [n]^d \setminus \emptyset} \phi_k^J(x, u^{(d)}) \frac{\partial}{\partial u_J^k} \tag{34}$$

with

$$\phi_k^J(x, u^{(d)}) = D_J\Big(\phi_k - \sum_{i=1}^{n} \xi^i u_i^k\Big) + \sum_{i=1}^{n} \xi^i u_{J \cup \{i\}}^k \tag{35}$$

and the total derivatives

$$D_{\{j_1, \ldots, j_l\}} = D_{j_1} \cdots D_{j_l} \quad \text{with} \;\; D_j = \frac{\partial}{\partial x_j} + \sum_{k=1}^{m} \sum_{J \in [n]^d} u_{J \cup \{j\}}^k \frac{\partial}{\partial u_J^k} \;\; . \tag{36}$$

Let us illustrate Theorem B.1 and these relatively lengthy formulas through a practical example.

Example B.2 [1d-heat equation]: Consider the heat equation

$$\mathcal{D}(x, t, u^{(2)}) = u_t - u_{xx} \tag{37}$$

with $n = 2$, $m = 1$, $p = 1$ and $d = 2$. Applying (34) and (32) to a generic vector field

$$\mathbf{v} = \xi(x,t,u)\frac{\partial}{\partial x} + \tau(x,t,u)\frac{\partial}{\partial t} + \phi(x,t,u)\frac{\partial}{\partial u} \tag{38}$$

we will obtain the infinitesimal criterion for symmetry groups of the heat equation as

$$\phi^t - \phi^{xx} = 0 \quad \text{whenever} \;\; u_t = u_{xx} \;\; . \tag{39}$$

To resolve the functions ϕ^t and ϕ^{xx}, we get by application of (35) and (36)

$$\begin{aligned} \phi^t &= D_t(\phi - \xi u_x - \tau u_t) + \xi u_{xt} + \tau u_{tt} = D_t\phi - u_x D_t\xi - u_t D_t\tau \\ &= \phi_t - \xi_t u_x + (\phi_u - \tau_t)u_t - \xi_u u_x u_t - \tau_u u_t^2 \end{aligned} \tag{40}$$

and

$$\begin{aligned} \phi^{xx} &= D_x^2(\phi - \xi u_x - \tau u_t) + \xi u_{xxx} + \tau u_{xxt} \\ &= D_x^2\phi - u_x D_x^2\xi - u_t D_x^2\tau - 2u_{xx}D_x\xi - 2u_{xt}D_x\tau \\ &= \phi_{xx} + (2\phi_{xu} - \xi_{xx})u_x - \tau_{xx}u_t + (\phi_{uu} - 2\xi_{xu})u_x^2 - 2\tau_{xu}u_x u_t - \xi_{uu}u_x^3 \\ &\quad - \tau_{uu}u_x^2 u_t + (\phi_u - 2\xi_x)u_{xx} - 2\tau_x u_{xt} - 3\xi_u u_x u_{xx} - \tau_u u_t u_{xx} - 2\tau_u u_x u_{xt} \;\; . \end{aligned} \tag{41}$$

Since $\phi^t = \phi^{xx}$ has to hold independently of the value of derivatives of u, we can compare them term-wise. When using in addition that $u_t = u_{xx}$, we get the following system of equations, which are often called the *determining equations*:

(i) $\phi_t = \phi_{xx}$

(ii) $-\xi_t = 2\phi_{xu} - \xi_{xx}$

(iii) $0 = \phi_{uu} - 2\xi_{xu}$

(iv) $0 = -\xi_{uu}$

(v) $\phi_u - \tau_t = -\tau_{xx} + \phi_u - 2\xi_x$

(vi) $-\xi_u = -2\tau_{xu} - 3\xi_u$

(vii) $0 = -\tau_{uu}$

(viii) $-\tau_u = -\tau_u$

(ix) $0 = -2\tau_x$

(x) $0 = -2\tau_u$

In order to find the most general infinitesimal generator $\mathbf{v}$ from (33), we have to solve this system, to determine the functions $\xi(x,t,u), \tau(x,t,u)$ and $\phi(x,t,u)$. From (ix) and (x), we get that $\tau = \tau(t)$ depends only on t. In combination with (vi), ξ will not depend on u. Hence, (v) takes the form $\tau_t = 2\xi_x$, which implies $\xi(x,t) = \frac{1}{2}\tau_t x + \gamma(t)$ for some function γ. Therefore, using (iii), ϕ will only linearly depend on u, i.e., $\phi(x,t,u) = \alpha(x,t) + \beta(x,t)u$ for some functions α, β. Hence, we have used every equation except (i) and (ii), which takes the form

(i) $\alpha_t = \alpha_{xx}$, $\beta_t = \beta_{xx}$

(ii) $-\frac{1}{2}\tau_{tt}x - \gamma_t = 2\beta_x$

By integrating (ii) w.r.t. x, we get $\beta(x,t) = -\frac{1}{8}\tau_{tt}x^2 - \frac{1}{2}\gamma_t x + \rho(t)$. Comparing with (i), we get $\tau_{ttt} = 0$, $\gamma_{tt} = 0$ and $\rho_t = -\frac{1}{4}\tau_{tt}$. Hence, $\tau(t) = 4c_6t^2 + 2c_4t + c_2$ is a quadratic function, $\gamma(t) = 2c_5t + c_1$ and $\rho(t) = -2c_6t + c_3$ are linear functions, with $c_1, \ldots, c_6 \in \mathbb{R}$. Thus, the general solution of the determining equations is given by

$$\xi = c_1 + c_4x + 2c_5t + 4c_6xt \tag{42}$$

$$\tau = c_2 + 2c_4t + 4c_6t^2 \tag{43}$$

$$\phi = (c_3 - c_5x - 2c_6t - c_6x^2)u + \alpha(x,t) \tag{44}$$

where $c_1, \ldots, c_6 \in \mathbb{R}$ and $\alpha(x,t)$ is any solution of the heat equation $\alpha_t = \alpha_{xx}$. Using the usual basis in $\mathbb{R}^6$, the Lie algebra is spanned by 6 vector fields

$$\begin{aligned} &\mathbf{v}_1 = \partial_x \qquad \mathbf{v}_2 = \partial_t \qquad \mathbf{v}_3 = u\partial_u \qquad \mathbf{v}_4 = x\partial_x + 2t\partial_t \\ &\mathbf{v}_5 = 2t\partial_x - xu\partial_u \qquad \mathbf{v}_6 = 4xt\partial_x + 4t^2\partial_t - (x^2 + 2t)u\partial_u \end{aligned} \tag{45}$$

and the subalgebra $\mathbf{v}_\alpha = \alpha(x,t)\partial_u$. △

Hence, by Theorem B.1 we can determine the infinitesimal generators of the symmetry group. In a next step, we have to relate these generators in the Lie algebra to the actual group elements. This connection is provided by the *exponential map* $\exp : \mathfrak{g} \to G$, whose action on an element $(x, u) \in M \subseteq X \times \mathcal{U}$ is described by

$$\exp(\vartheta \mathbf{v}) \circ (x, u) = \sum_{k \geq 0} \frac{\vartheta^k}{k!} \mathbf{v}^k \circ (x, u) \ . \tag{46}$$

Let us illustrate the application of the exponential map by continuing the previous example.

Example B.3 [Continuation of Example B.2]: To compute the one-parameter group G_1 associated to the generator $\mathbf{v}_1$ from (45) we have to apply $\mathbf{v}_1$ to a graph of a function, i.e., to an element $(x, t, u) \in M$. We get $\mathbf{v}_1 \circ (x, t, u) = \partial_x \circ (x, t, u) = (1, 0, 0)$ and $\mathbf{v}^k \circ (x, t, u) = (0, 0, 0)$ for any $k > 1$. Thus, the group G_1 is acting by

$$G_1 : (x, t, u) \mapsto \exp(\vartheta \mathbf{v}_1) \circ (x, t, u) = (x, t, u) + (1, 0, 0)\vartheta = (x + \vartheta, t, u) \tag{47}$$

which is nothing else than a translation in the x coordinate. Similarly, we get a translation in time

$$G_2 : (x, t, u) \mapsto \exp(\vartheta \mathbf{v}_2) \circ (x, t, u) = (x, t + \vartheta, u) \ . \tag{48}$$

To compute the group G_3 note, that $\mathbf{v}_3^k \circ u = u$ for all $k \in \mathbb{N}$. Hence, the generator $\mathbf{v}_3$ leads to a scaling invariance

$$G_3 : (x, t, u) \mapsto \exp(\vartheta \mathbf{v}_3) \circ (x, t, u) = \Big(x, t, u \sum_{k \geq 0} \frac{\vartheta^k}{k!}\Big) = (x, t, e^{\vartheta} u) \ . \tag{49}$$

Quite similar is the computation for $\mathbf{v}_4$, and with $\mathbf{v}_4^k \circ x = x$ and $\mathbf{v}_4^k \circ t = 2^k t$ we obtain

$$G_4 : (x, t, u) \mapsto \exp(\vartheta \mathbf{v}_4) \circ (x, t, u) = (e^{\vartheta} x, e^{2\vartheta} t, u) \ . \tag{50}$$

To compute the group action of G_5, note that

$$\mathbf{v}_5^k \circ u = u \sum_{0 \leq m \leq k/2} \frac{k!}{m!(k - 2m)!} (-x)^{k-2m} (-t)^m \ , \tag{51}$$

which can be shown by induction including a distinction between even and odd k. Including that relation in the series of (46) we obtain

$$G_5 : (x, t, u) \mapsto \exp(\vartheta \mathbf{v}_5) \circ (x, t, u) = \Big(x + 2\vartheta t, t, u e^{-\vartheta x - \vartheta^2 t}\Big) \ . \tag{52}$$

To find the action of the group G_6, we will use another useful property of the exponential map, which is

$$A \circ \exp(B) \circ A^{-1} = \exp(A \circ B \circ A^{-1}) \tag{53}$$

for any functions A, B with appropriate domains and codomains and where $A \circ A^{-1} = \mathbf{1}$. This relation can directly be shown by the definition (46). In particular, the so-called inversion map

$$I(x, t) = \Big(\frac{x}{t^2 - x^2}, \frac{t}{t^2 - x^2}\Big) \tag{54}$$

has the property $I \circ I = \mathbf{1}$. As we can write $\exp(\vartheta \mathbf{v}_6) = I \circ \exp(-\vartheta \partial_x) \circ I$ we will find

$$G_6 : (x, t, u) \mapsto \exp(\vartheta \mathbf{v}_6) \circ (x, t, u) = \Big(\frac{x}{1 - 4\vartheta t}, \frac{t}{1 - 4\vartheta t}, u\sqrt{1 - 4\vartheta t}\, e^{-\frac{\vartheta x^2}{1 - 4\vartheta t}}\Big) \ . \tag{55}$$

The last generator $\mathbf{v}_\alpha$ will result in the action

$$G_\alpha : (x, t, u) \mapsto (x, t, u + \vartheta\, \alpha(x, t)) \ . \tag{56}$$

△

In summary, we have now determined the group actions on the graph of a function $\Gamma_f \subset X \times \mathcal{U}$, i.e., the action on every point (x, u) of that graph. In a final step, we would like to transfer this action onto the function itself. For simplicity, we will focus on the case where the action of the group element $g \in G$ takes the form $(\tilde{x}, \tilde{u}) = g \circ (x, u) = (\Xi_g(x), \Phi_g(x, u))$, i.e., that the action of g on x will not depend on u. In this particular case, we have $x = \Xi_g(\tilde{x})^{-1} = \Xi_{g^{-1}}(\tilde{x})$, and therefore

$$\tilde{f}(\tilde{x}) = \Phi_g(\Xi_{g^{-1}}(\tilde{x}), f(\Xi_{g^{-1}}(\tilde{x}))) \ . \tag{57}$$

For the general case, we refer to (Olver, 1993, sec. 2.2).

Example B.4 [Continuation of Example B.3]: By using (57), we can relate the group actions from Example B.3 to transformations of functions. For example, for G_1 we have $\Xi_{g_{1,\vartheta}}(x) = x + \vartheta$ and $\Phi_{g_{1,\vartheta}}(x, u) = u$. With $\Xi_{g_{1,\vartheta}^{-1}}(x) = \Xi_{g_{1,-\vartheta}}(x) = x - \vartheta$, we get the transformation

$$T^1_\vartheta : f(x, t) \mapsto f(x - \vartheta, t) \tag{58}$$

as expected. By applying (57) to the other groups $G_2, \dots, G_6, G_\alpha$, we obtain the results presented in Section 3.2

$$T^2_\vartheta : f(x, t) \mapsto f(x, t - \vartheta) \ , \tag{59}$$

$$T^3_\vartheta : f(x, t) \mapsto e^\vartheta f(x, t) \ , \tag{60}$$

$$T^4_\vartheta : f(x, t) \mapsto f(e^{-\vartheta} x, e^{-2\vartheta} t) \ , \tag{61}$$

$$T^5_\vartheta : f(x, t) \mapsto e^{-\vartheta x + \vartheta^2 t} f(x - 2\vartheta t, t) \ , \tag{62}$$

$$T^6_\vartheta : f(x, t) \mapsto \frac{1}{\sqrt{1 + 4\vartheta t}} \exp\left(\frac{-\vartheta x^2}{1 + 4\vartheta t}\right) f\left(\frac{x}{1 + 4\vartheta t}, \frac{t}{1 + 4\vartheta t}\right) \ , \tag{63}$$

$$T^\alpha_\vartheta : f(x, t) \mapsto f(x, t) + \vartheta\, \alpha(x, t) \ . \tag{64}$$

for any $\vartheta \in \mathbb{R}$, such that the functions are defined and an arbitrary solution $\alpha(x, t)$ of the heat equation. Hence, for any solution $u = f(x, t)$ of the heat equation (37), applying any of the aforementioned transformations produces another possible solution. As an example, for the constant solution $f(x, t) \equiv 1$ and after applying the transformation T^6_ϑ, one obtains the function $u = \frac{1}{\sqrt{1+4\vartheta t}} \exp\left(\frac{-\vartheta x^2}{1+4\vartheta t}\right)$ that also solves the heat equation, whenever this function is defined, i.e. for $1 + 4\vartheta t \neq 0$.

While the transformations $T^1, \dots, T^4$ can be found by carefully observing the differential equation (37), T^5 and T^6 are far from being trivial to be obtained by reading the PDE. △

Thus, we have a framework that, in principle, allows us to compute a symmetry group for any PDE system. For PDE systems with a certain additional property, we can give even a stronger statement. According to (Olver, 1993, Def. 2.70), a system of PDEs $\mathcal{D}$ is called *local solvable* at a point $(x_0, u_0^{(d)}) \in X \times \mathcal{U}^{(d)}$, if there is a smooth solution $u = f(x)$ for x in a neighbourhood of x_0, with $u_0^{(d)} = \mathrm{pr}^{(d)} f(x_0)$. The whole PDE system $\mathcal{D}$ is called locally solvable, if this property holds for any point $(x_0, u_0^{(d)})$ satisfying $\mathcal{D}(x_0, u_0^{(d)}) = 0$. For a locally solvable system, we can give a sufficient and necessary description of symmetry groups:

Theorem B.5 (Infinitesimal criterion (see Olver, 1993, thm. 2.71))**.** *Let $\mathcal{D}$ be a locally solvable system of PDEs with maximal rank defined over $M \subseteq X \times U$ and let G be a local group of transformations acting on M. G is a symmetry group of $\mathcal{D}$,* if and only if

$$\mathrm{pr}^{(d)}\, \mathbf{v}[\mathcal{D}(x, u^{(d)})] = 0, \qquad \textit{whenever } \mathcal{D}(x, u^{(d)}) = 0 \tag{65}$$

holds for any infinitesimal generator $\mathbf{v}$ *of* G.

Thus, for locally solvable systems, we can derive *all* symmetry groups of the PDE system with the aforementioned framework. This ability to characterize the complete symmetry group of the solution manifold is precisely what renders Lie symmetries so powerful for our approach.

We would like to conclude this section by recapping all the steps to compute the symmetry group:

0. Check maximal rank and local solvability of $\mathcal{D}$.

1. Write an ansatz for a vector field $\mathbf{v}$ on $X \times \mathcal{U}$ according to (33) and compute its prolongation by (34), (35) and (36).
2. Derive the infinitesimal criterion Theorems B.1 and B.5, i.e., the prolongation of $\mathbf{v}$ applied on the PDE should vanish whenever the PDE is satisfied.
3. By comparing the coefficients of the infinitesimal criterion, one obtains an overdetermined system of simple PDEs ("determining equations").
4. Solve the determining equations by elementary methods, to get the most general generators $\mathbf{v}$ and choose a basis of the subspace generated by $\mathbf{v}$.
5. Derive one-parameter Lie groups from infinitesimal generators by the exponential map $g_\vartheta \circ (x, u) = \exp(\vartheta \mathbf{v}) \circ (x, u)$.
6. Derive the transformation maps from these group actions by (57).

There are several implementations available, which will solve this task step by step. We refer to (Steinberg & de Melo Marinho Junior, 2014; Hereman, 1996; Butcher et al., 2003; Champagne et al., 1991), which give an overview about different available implementations.

C. Experimental Details

C.1. Reference Solution

In order to compare the predictions of LIESOLVER, we compute reference ground-truth solutions by expanding the initial data in a sine basis according to (Boyce, 2016). For the heat equation with Dirichlet BC, we set $b_L = f(0, 0)$ and $b_R = f(1, 0)$ and define $w(x) = b_L + (b_R - b_L)\, x$ so that $v_0(x) = f(x, 0) - w(x)$ satisfies homogeneous Dirichlet conditions. We sample v_0 at the interior points $x_m = \frac{m}{M+1}$ for $m = 1, \dots, M$ and obtain the coefficients A_m via orthonormal projection onto $\sin(m\pi x)$. The solution is

$$f_{\text{ref}}(x, t) = w(x) + \sum_{m=1}^{M} A_m \, \sin\big(m\pi x\big) \, e^{-m^2\pi^2 t}. \tag{66}$$

For the wave equation with homogeneous Dirichlet boundaries, we sample $f(x, 0)$ and $f_t(x, 0)$ at the same interior points x_m and compute A_m and B_m by orthonormal projection onto $\sin(m\pi x)$. The solution is then given by

$$f_{\text{ref}}(x, t) = \sum_{m=1}^{M} \Big[A_m \cos\big(m\pi t\big) + \frac{B_m}{\omega_m} \sin\big(m\pi t\big)\Big] \sin\big(m\pi x\big). \tag{67}$$

C.2. PINN Baseline Setup

We implement our PINN baseline using the DeepXDE framework (Lu et al., 2021a) with a PyTorch backend. The model $f_{\text{PINN}} : \mathbb{R}^2 \to \mathbb{R}$ is a fully connected NN with $\tanh$ activation and Glorot normal initialization. The number of layers and width are specified per case in Table 4, following recommendations from (Wang et al., 2023).

Using the operators $\mathcal{D}$ and $\mathcal{B}_k$ from Section 3.1, the PINN loss is

$$\mathcal{L}_{\text{res}}(\boldsymbol{w}) = \frac{1}{N_r} \sum_{x_k \in \mathcal{X}_r} \big(\mathcal{D}[f_{\text{PINN}}](x_k; \boldsymbol{w})\big)^2, \tag{68}$$

$$\mathcal{L}_{\text{IBC}}(\boldsymbol{w}) = \frac{1}{L} \sum_{k=1}^{r} \sum_{x_j \in \mathcal{X}_k} \big(\mathcal{B}_k[f_{\text{PINN}}](x_j; \boldsymbol{w})\big)^2, \tag{69}$$

$$\mathcal{L}(\boldsymbol{w}) = \mathcal{L}_{\text{res}}(\boldsymbol{w}) + w_{\text{IBC}} \, \mathcal{L}_{\text{IBC}}(\boldsymbol{w}) \ , \tag{70}$$

where $\mathcal{X}_r \subset \Omega$ are domain collocation points and $\mathcal{X}_k \subset \Gamma_k$ are boundary points. For the wave equation, $\mathcal{L}_{\text{IBC}}$ includes the zero initial velocity condition $\partial_t f|_{t=0} = 0$ as an additional boundary term. The weight w_{IBC} is tuned per case to balance the loss components (see Table 4).

We sample 4000 domain points and 1000–1500 boundary points using Hammersley sequences. Training proceeds with Adam for 1000 iterations followed by L-BFGS until convergence or optimizer stall. All experiments were run on the same

hardware[10] as LIESOLVER. Reported runtimes measure wall-clock time from initialization to convergence, excluding data generation.

C.3. IBC Data Specifications

Heat Equation We consider the domain $(x, t) \in [0, 1] \times [0, 0.1]$. The boundary conditions are fixed to the constant values of the initial profile at the boundaries: $f(0, t) = f_0(0)$ and $f(1, t) = f_0(1)$. We employ five distinct initial profiles $f(x, 0) = f_0(x)$, visualized in the top row of Figure 7:

$$\textit{Poly}: \quad f_0(x) = x^2 + x^3 - x^5 + x^7 \tag{71}$$

$$\textit{Gauss}: \quad f_0(x) = e^{-5(x-0.5)^2} \tag{72}$$

$$\textit{Sine}: \quad f_0(x) = \sin(4\pi x) \tag{73}$$

$$\textit{Sine Mix}: \quad f_0(x) = 0.5\sin(2\pi x) - 0.2\sin(4\pi x) + 0.7\sin(12\pi x) \tag{74}$$

$$\textit{Step}: \quad f_0(x) = 0.5\Big[\tanh\big(500(x-0.4)\big) - \tanh\big(500(x-0.6)\big)\Big] \tag{75}$$

To generate the training dataset $\mathcal{X}$, we sample points using a low-discrepancy Halton sequence to ensure uniform coverage of the initial and boundary manifolds. Specifically, we sample 1000 points at $t = 0$, 1000 points at each boundary $x = 0$ and $x = 1$, resulting in a total dataset size of $L = 3000$. We generate a test set of 3000 uniformly sampled points on the IBCs and a domain set of 10000 uniformly sampled points.

Wave Equation We consider the domain $(x, t) \in [0, 1] \times [0, 1]$. We apply homogeneous Dirichlet boundary conditions $f(0, t) = f(1, t) = 0$ and a zero initial velocity condition $f_t(x, 0) = 0$. The initial displacement profiles $f(x, 0) = f_0(x)$ mirror the complexity of the heat equation cases (see bottom row of Figure 7):

$$\textit{Gauss}: \quad f_0(x) = e^{-50(x-0.5)^2} \tag{76}$$

$$\textit{Gauss Mix}: \quad f_0(x) = 0.7e^{-100(x-0.4)^2} + e^{-500(x-0.7)^2} \tag{77}$$

$$\textit{Sine}: \quad f_0(x) = \sin(4\pi x) \tag{78}$$

$$\textit{Sine Mix}: \quad f_0(x) = 0.5\sin(2\pi x) - 0.2\sin(4\pi x) + 0.7\sin(12\pi x) \tag{79}$$

$$\textit{Step}: \quad f_0(x) = 0.5\Big[\tanh\big(500(x-0.4)\big) - \tanh\big(500(x-0.6)\big)\Big] \tag{80}$$

Training data is similarly sampled via the Halton sequence. To prevent overfitting we increase the sampling density relative to the heat equation. We use 1500 points for the initial displacement, 1000 points for the initial velocity, and 1250 points for each boundary, resulting in a total dataset size of $L = 5000$. We generate a test set of 5000 uniformly sampled points on the IBCs and a domain set of 10000 uniformly sampled points.

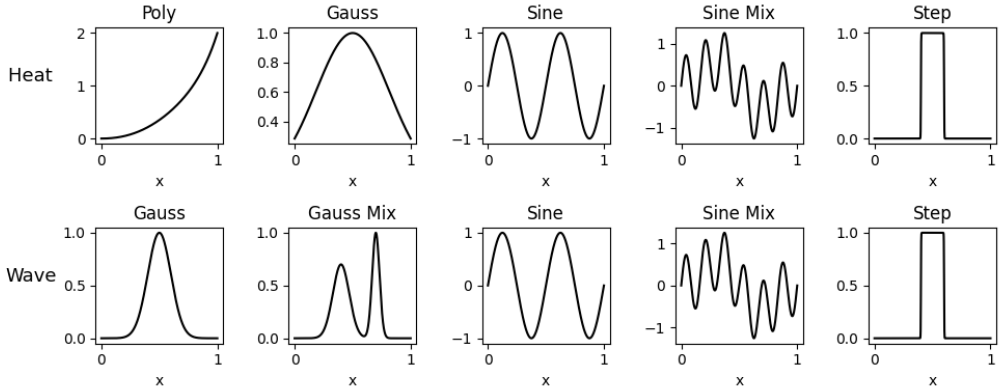


Figure 7. Initial conditions (solution shapes at $t = 0$) for the test cases for the heat (top row) and the wave (bottom row) equations.

[10] Intel Core Ultra 7 165H CPU with 11 cores and 16 GiB RAM.

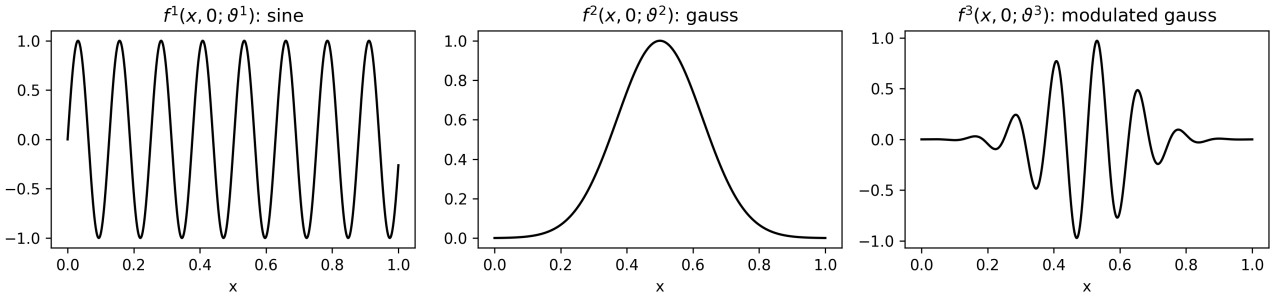


Figure 8. Representative brick families for one-dimensional heat equation at $t = 0$: f^1 (sine), f^2 (Gaussian), and f^3 (Gaussian modulated by a sine) with illustrative parameter choices.

Brick specification details Figure 8 illustrates the variety of brick solutions at $t = 0$ constructed to solve the one-dimensional heat equation. The choice of parameter bounds and sampling strategies for the Lie symmetries is crucial for model performance; ranges must be physics-informed to ensure that the generated features lie within the physical domain and cover the relevant scales. E.g. we employ log-uniform sampling for scaling parameter of transformation (9) to capture features across multiple orders of magnitude.

For the heat equation, we utilize the following specifications:

- f^1 (sine mode): $T^4_{\vartheta_2} \cdot T^1_{\vartheta_1} \circ (e^{-t} \sin x)$, with phase shift $\vartheta_1 \sim \mathcal{U}(0, \pi)$ and frequency $\vartheta_2 \sim \mathcal{U}(0, 100)$.
- f^2 (Gaussian): $T^1_{\vartheta_2} \cdot T^6_{\vartheta_1} \circ (1)$, with position $\vartheta_2 \sim \mathcal{U}(0, 1)$ and shape parameter $\vartheta_1 \sim \text{Log}\mathcal{U}(10^{-1}, 10^6)$.
- f^3 (Modulated Gaussian): $T^1_{\vartheta_4} \cdot T^6_{\vartheta_3} \cdot T^4_{\vartheta_2} \cdot T^1_{\vartheta_1} \circ (e^{-t} \sin x)$, combining the bounds of the previous families to generate localized oscillatory features.

For the wave equation, we specify:

- f^1 (Standing wave): $T^2_{\vartheta_2} \cdot T^1_{\vartheta_1} \circ (\sin x \cos t)$, with phase $\vartheta_1 \sim \mathcal{U}(0, \pi)$ and frequency scaling $\vartheta_2 \sim \mathcal{U}(0, 100)$.
- f^2 (Travelling wave): $T^1_{\vartheta_2} \cdot T^2_{\vartheta_1} \circ (e^{-(x+t)^2} + e^{-(x-t)^2})$, with translation $\vartheta_2 \sim \mathcal{U}(-1, 2)$ to allow waves to enter from boundaries, and width scaling $\vartheta_1 \sim \mathcal{U}(0, 1000)$.

Table 3. Median LIESOLVER runs for the one-dimensional heat and wave equations, reporting metrics, runtime and model complexity.

PDE	IBC Type	MSE IBC	L2RE domain	Runtime [s]	# Bricks	# Parameters
Heat	*Poly*	$8.2 \cdot 10^{-7}$	$4.4 \cdot 10^{-4}$	34	31	86
	Gauss	$2.4 \cdot 10^{-8}$	$2.0 \cdot 10^{-4}$	11	15	40
	Sine	$2.4 \cdot 10^{-8}$	$4.6 \cdot 10^{-4}$	4	5	12
	Sine Mix	$2.0 \cdot 10^{-8}$	$3.5 \cdot 10^{-4}$	10	15	38
	Step	$9.1 \cdot 10^{-7}$	$1.3 \cdot 10^{-3}$	310	78	230
Wave	*Gauss*	$4.9 \cdot 10^{-8}$	$1.1 \cdot 10^{-3}$	8	15	30
	Gauss Mix	$7.8 \cdot 10^{-7}$	$5.0 \cdot 10^{-3}$	41	41	82
	Sine	$1.1 \cdot 10^{-7}$	$8.1 \cdot 10^{-4}$	5	9	18
	Sine Mix	$4.6 \cdot 10^{-7}$	$1.7 \cdot 10^{-3}$	4	7	14
	Step	$9.4 \cdot 10^{-7}$	$4.0 \cdot 10^{-3}$	217	91	182

Table 4. Median PINN runs for the one-dimensional heat and wave equations, reporting metrics, runtime, and model configuration.

PDE	IBC Type	MSE (IBC)	L2RE (domain)	Runtime [s]	Training iterations	Network	IBC weight
Heat	*Poly*	$9.0 \cdot 10^{-7}$	$4.8 \cdot 10^{-4}$	137	4472	4×50	10^3
	Gauss	$3.2 \cdot 10^{-7}$	$4.9 \cdot 10^{-4}$	123	4554	4×50	10^3
	Sine	$7.8 \cdot 10^{-7}$	$5.0 \cdot 10^{-3}$	183	5238	4×50	10^3
	Sine Mix	$4.6 \cdot 10^{-6}$	$1.9 \cdot 10^{-2}$	1716	10674	6×100	10^5
	Step	$1.8 \cdot 10^{-3}$	$2.2 \cdot 10^{-1}$	440	4005	6×100	10^5
Wave	*Gauss*	$1.5 \cdot 10^{-7}$	$1.6 \cdot 10^{-3}$	1119	8321	4×50	10^3
	Gauss Mix	$2.7 \cdot 10^{-6}$	$8.2 \cdot 10^{-3}$	2456	17864	4×50	10^3
	Sine	$2.0 \cdot 10^{-7}$	$8.5 \cdot 10^{-4}$	1172	8744	4×50	10^3
	Sine Mix	$8.2 \cdot 10^{-5}$	$1.8 \cdot 10^{-2}$	11008	22000	6×100	10^3
	Step	$7.2 \cdot 10^{-5}$	$4.5 \cdot 10^{-2}$	909	7786	4×50	10^3

D. Experimental Results

We present the detailed quantitative results for all experimental cases in Table 3 for LIESOLVER and Table 4 for the baseline PINN. For reference, a standard finite difference method (FDM) (LeVeque, 2007) solves these benchmarks in under a second with comparable accuracy.

For completeness, we also evaluated additional neural baselines: a hard-constrained PINN (Lu et al., 2021b), which modifies the network output to satisfy IBCs exactly by construction, and a physics-informed Kolmogorov-Arnold network (Liu et al., 2025). Both methods performed comparably to or worse than the standard PINN baseline in accuracy and runtime on our considered benchmarks. We therefore focus our comparison on the standard PINN only.

In the following, we discuss the LIESOLVER optimization behaviour and qualitative results for the specific ICs.

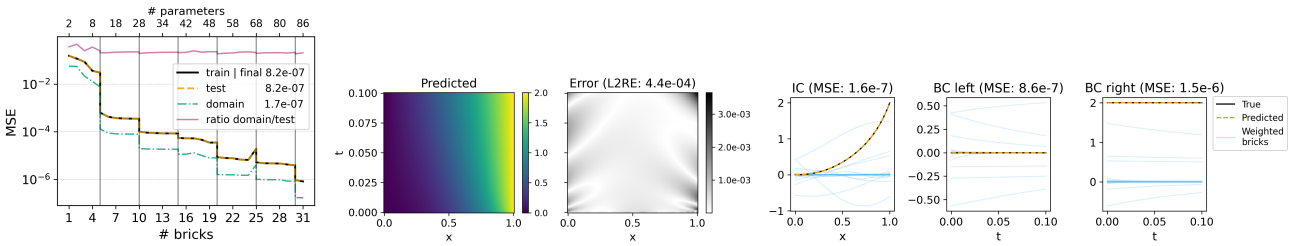
D.1. LieSolver Results

We analyze the performance of LIESOLVER across different ICs, highlighting optimization specifics.

Heat *Poly*: The fit progress shown in Figure 9a illustrates a fallback of the greedy structural learning stage. Around the 25th brick addition, we observe a temporary worsening of the MSE. This occurs because the greedy initialization selects a brick from a finite random pool based on cosine similarity with the current residual. While this candidate is locally optimal with respect to the pool, its initial addition to the linear combination may temporarily disrupt the global fit due to the irregular residual profile. However, the subsequent global parameter refinement stage successfully resolves this, realigning the parameters to minimize the global loss and allowing the model to monotonically converge to the target MSE with 31 bricks.

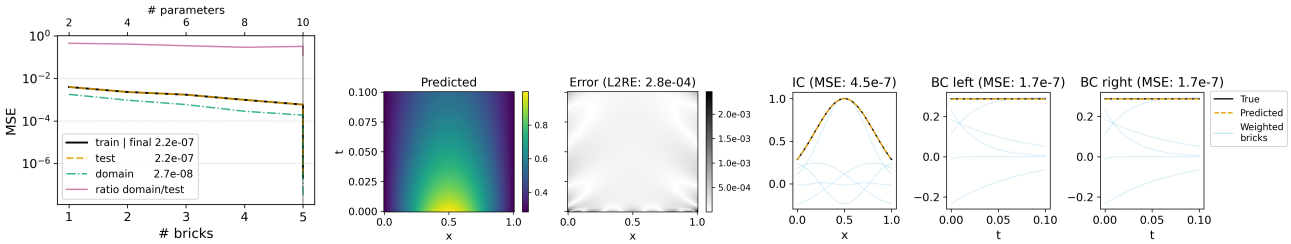
Heat *Gauss*: While the aggregate results in Table 3 utilized our standard hyperparameter setting (`nfev`=4), we illustrate a specific run in Figure 10 where the NLLS refinement budget was increased to `nfev`=40. This demonstrates the trade-off between structural search and parameter optimization: with deeper refinement, the optimizer locates a high-quality local minimum more effectively, allowing LIESOLVER to satisfy the tolerance with only 5 bricks (compared to 15 bricks in the median run in the main text). The decomposition in Figure 10c shows the solution is comprised of a primary Gaussian brick supplemented by two pairs of symmetric sine waves to correct boundary deviations.

Heat *Sine* and *Sine Mix*: As the set of brick families contains sine-generated bricks, LIESOLVER handles these cases (Figures 11 and 12) with high efficiency. Technically, a single sine brick suffices for the *Sine* case due to the rapid dissipation



(a) Fit progress. *(b)* 2D domain fit. *(c)* IBC fit.

Figure 9. LIESOLVER results for heat equation with *Poly* IC.



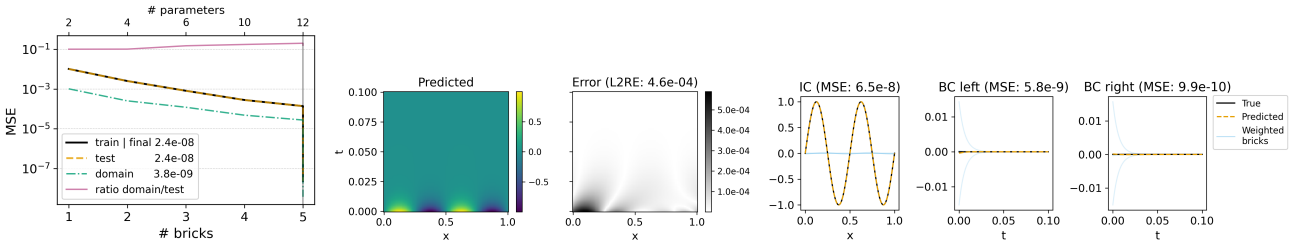
(a) Fit progress. *(b)* 2D domain fit. *(c)* IBC fit.

Figure 10. LIESOLVER results for heat equation with *Gauss* IC. Table 3 shows the result for run from the main text.

of boundary values under homogeneous Dirichlet conditions. However the obtained results confirm that our generalized training pipeline robustly identifies these simple solutions without instability, fitting *Sine* with 5 bricks and *Sine Mix* with 15 bricks.

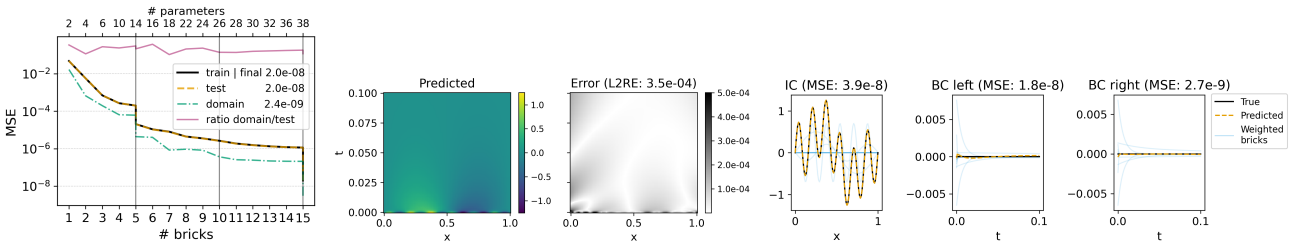
Heat *Step*: This profile (Figure 13) represents the most significant challenge due to the sharp jump. While a coarse fit (10^{-5} MSE) is achieved in approximately 100 seconds and 40 bricks, reaching the target 10^{-6} requires over 300 seconds and nearly 80 bricks. Similar to the *Poly* case, the fit history (Figure 13a) displays non-monotonic behaviour during greedy initialization, which is eventually corrected with global refinement upon increase of model expressivity.

While PINNs perform adequately on smooth profiles, they exhibit significant failure modes on the *Step* IC. As seen in Figure 22, the transition from an extremely sharp initial condition to a rapidly smoothening solution in the domain proves difficult for the network to resolve. Even with increased network capacity, the PINN cannot resolve the transition, leading to high metrics.



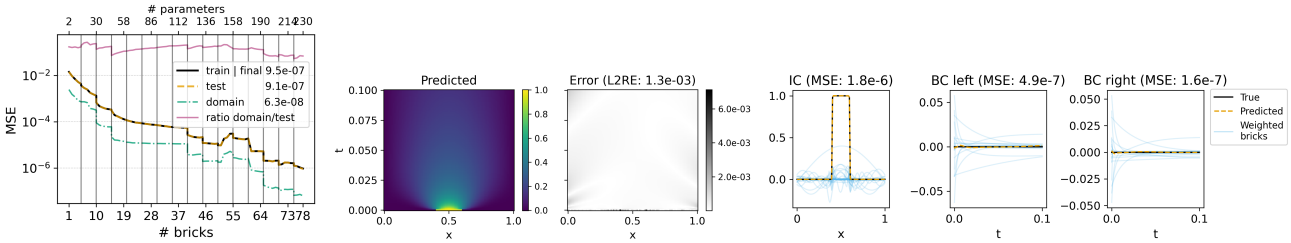
(a) Fit progress. *(b)* 2D domain fit. *(c)* IBC fit.

Figure 11. LIESOLVER results for heat equation with *Sine* IC.



(a) Fit progress. *(b)* 2D domain fit. *(c)* IBC fit.

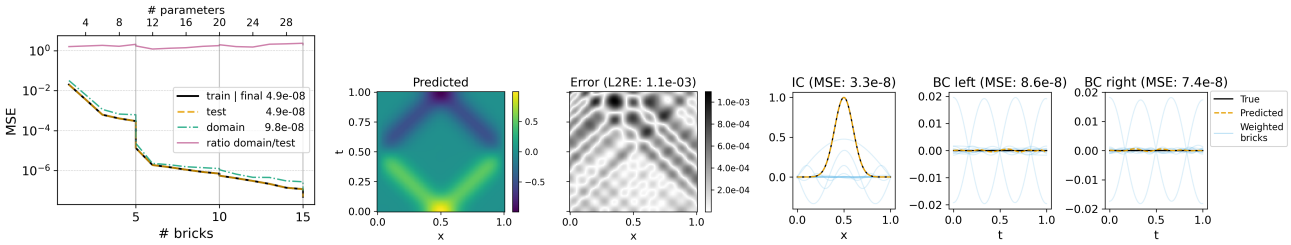
Figure 12. LIESOLVER results for heat equation with *Sine Mix* IC.



(a) Fit progress. *(b)* 2D domain fit. *(c)* IBC fit.

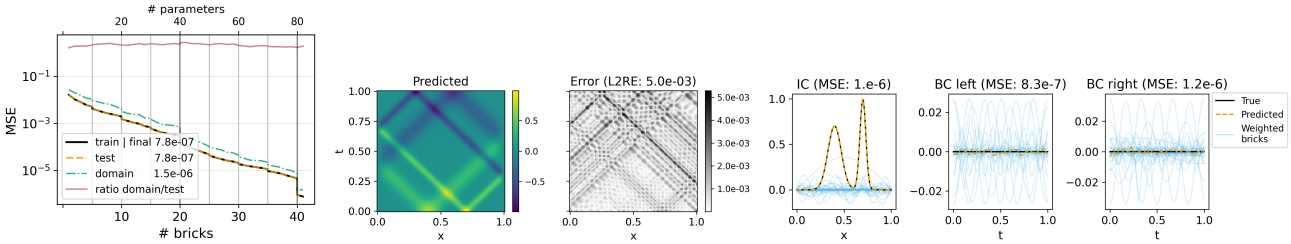
Figure 13. LIESOLVER results for heat equation with *Step* IC.

Wave Equation Cases: For the wave equation, we observe a consistent structural learning pattern across the *Gauss*, *Gauss Mix*, *Sine*, and *Sine Mix* cases (Figures 14 to 17). The greedy initialization predominantly spawns sine-wave bricks f^1 first, as these patterns naturally fit the oscillatory nature of the wave equation's solutions, before refining the fit with localized Gaussian corrections where necessary.



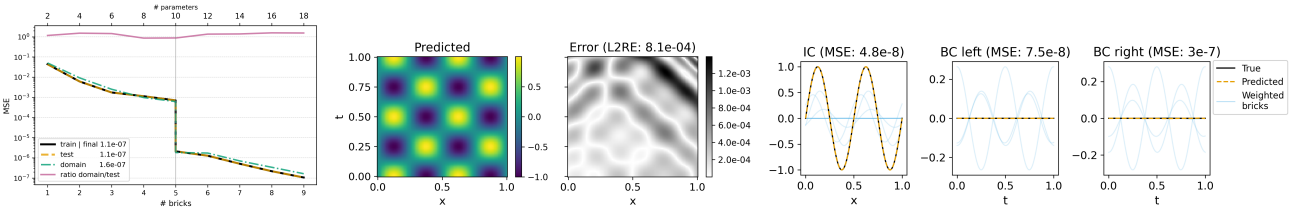
(a) Fit progress. *(b)* 2D domain fit. *(c)* IBC fit.

Figure 14. LIESOLVER results for wave equation with *Gauss* IC.



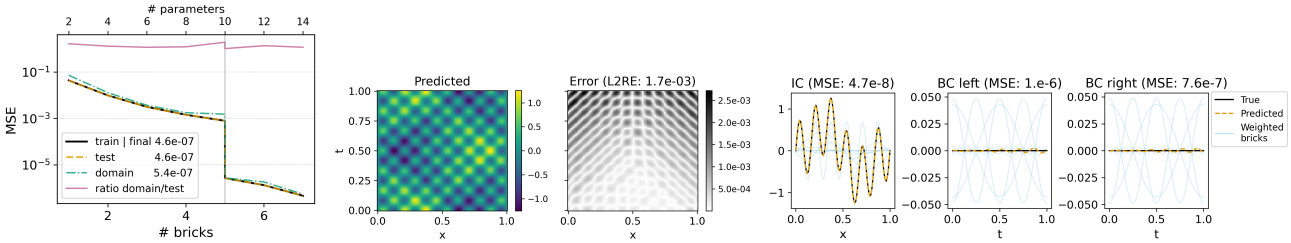
(a) Fit progress. *(b)* 2D domain fit. *(c)* IBC fit.

Figure 15. LIESOLVER results for wave equation with *Gauss Mix* IC.



(a) Fit progress. *(b)* 2D domain fit. *(c)* IBC fit.

Figure 16. LIESOLVER results for wave equation with *Sine* IC.

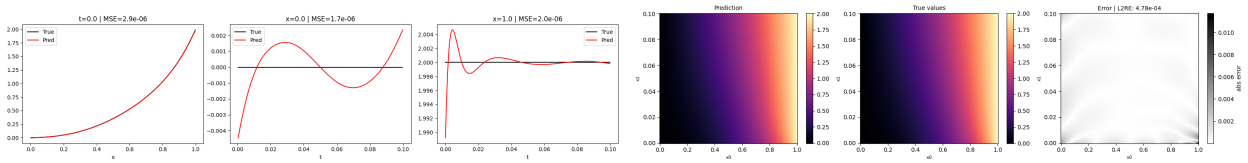


(a) Fit progress. *(b)* 2D domain fit. *(c)* IBC fit.

Figure 17. LIESOLVER results for wave equation with *Sine Mix* IC.

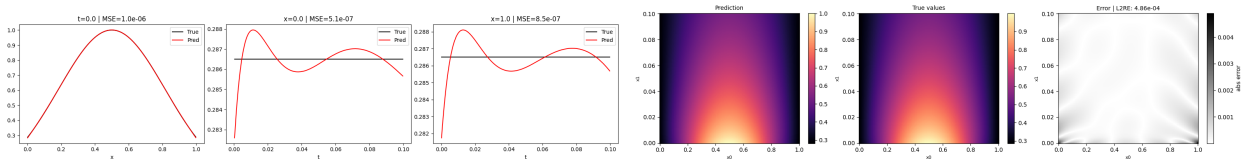
D.2. PINN Results

For completeness, we include the visual results of the PINN baselines in Figure 18 through Figure 27.



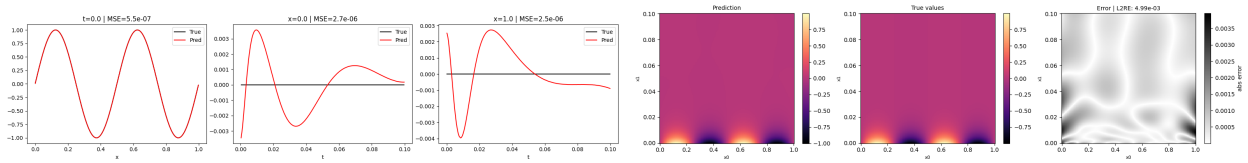
(a) IBC fit. *(b)* 2D domain fit.

Figure 18. PINN results for heat equation with *Poly* IC.



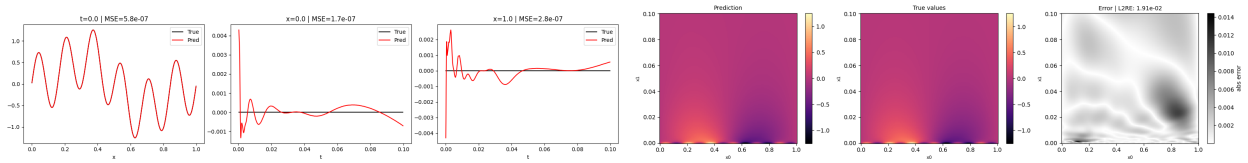
(a) IBC fit. *(b)* 2D domain fit.

Figure 19. PINN results for heat equation with *Gauss* IC.



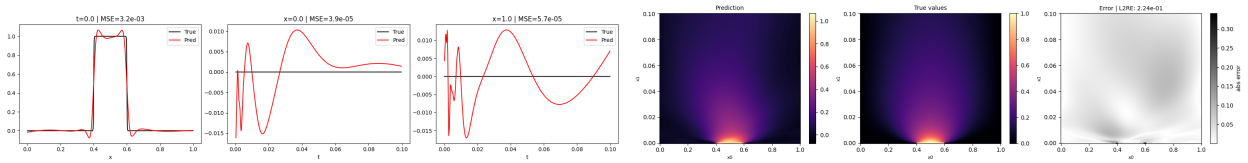
(a) IBC fit. *(b)* 2D domain fit.

Figure 20. PINN results for heat equation with *Sine* IC.



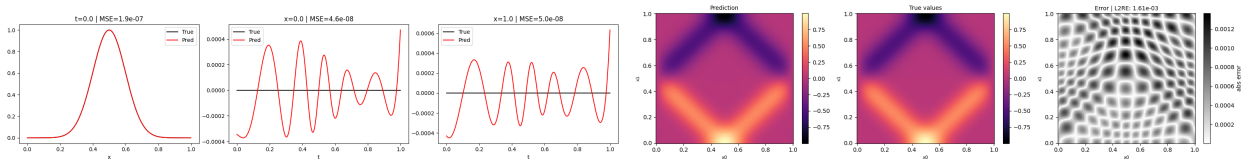
(a) IBC fit. *(b)* 2D domain fit.

Figure 21. PINN results for heat equation with *Sine Mix* IC.



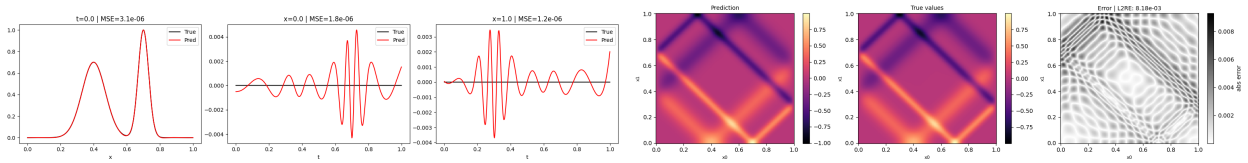
(a) IBC fit. *(b)* 2D domain fit.

Figure 22. PINN results for heat equation with *Step* IC.



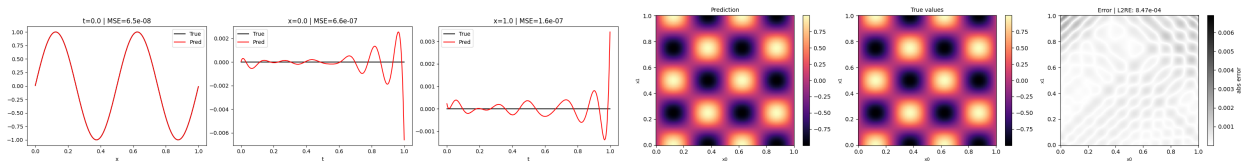
(a) IBC fit. *(b)* 2D domain fit.

Figure 23. PINN results for wave equation with *Gauss* IC.



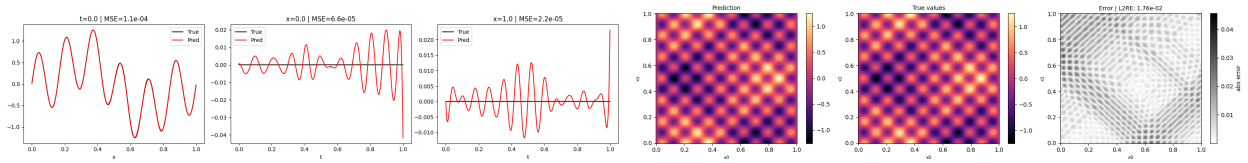
(a) IBC fit. *(b)* 2D domain fit.

Figure 24. PINN results for wave equation with *Gauss Mix* IC.



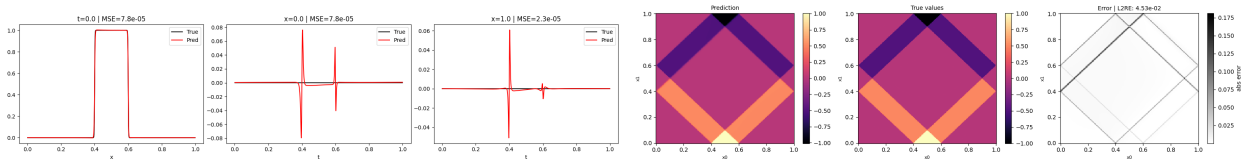
(a) IBC fit. *(b)* 2D domain fit.

Figure 25. PINN results for wave equation with *Sine* IC.



(a) IBC fit. *(b)* 2D domain fit.

Figure 26. PINN results for wave equation with *Sine Mix* IC.



(a) IBC fit. *(b)* 2D domain fit.

Figure 27. PINN results for wave equation with *Step* IC.