Lie Point Symmetry and Physics Informed Neural Networks

Tara Akhound-Sadegh [1 2] **Laurence Perreault-Levasseur** [3 2 4] **Johannes Brandstetter** [5] **Max Welling** [6 5] **Siamak Ravanbakhsh** [1 2]

Abstract

Physics-informed neural networks (PINNs) are computationally efficient alternatives to traditional partial differential equation (PDE) solvers. However, their reliability is dependent on the accuracy of the trained neural network. In this work, we introduce a mechanism for leveraging the symmetries of a given PDE to improve PINN performance. In particular, we propose a loss function that informs the network about Lie point symmetries, similar to how traditional PINN models try to enforce the underlying PDE. Intuitively, our symmetry loss ensures that infinitesimal generators of the Lie group preserve solutions of the PDE. Effectively, this means that once the network learns a solution, it also learns the neighbouring solutions generated by Lie point symmetries. Our results confirm that Lie point symmetries of the respective PDEs are an effective inductive bias for PINNs and can lead to a significant increase in sample efficiency.

1. Introduction

As machine learning advances accelerate a data-driven approach to science, the role of ML for solving differential equations is also becoming more pronounced. Since traditional numerical solvers can be prohibitively expensive, learning to solve PDEs, as an alternative to existing numerical solvers, can significantly impact various areas of science, ranging from biology to climate science to cosmology, (Wang et al., 2020a; Kochkov et al., 2021; Kashinath et al., 2021).

Like any other machine learning problem, learning to solve

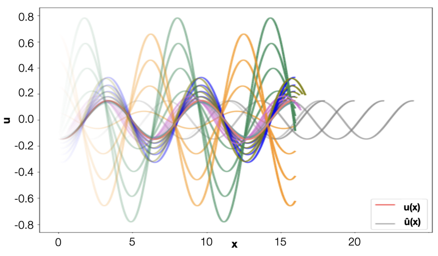

Figure 1. All solutions of the harmonic oscillator ODE can be reached via symmetry transformations of a single solution. Its Lie-point symmetry group is $SL(3)$, corresponding to eight one-parameter subgroups transforming a given solution, identified using a different colour in this figure. While, in general, Lie-point symmetries form more than one orbit, their use can still significantly narrow down the solution space.

PDEs can benefit from inductive biases to boost sample efficiency and generalization capabilities. As such, the PDE itself and its respective Lie point symmetries are two natural inductive biases for neural PDE solvers. While Physics-Informed Neural Networks (PINNs) constrain the solution network to satisfy the underlying PDE via the PINN loss terms (Raissi et al., 2019), the latter objective of using Lie point symmetries has so for been used for data augmentation (Brandstetter et al., 2022a), and to construct equivariant neural operators by incorporating single symmetry groups (Wang et al., 2021a). In this work, we show how to create additional loss functions for PINNs that locally enforce Lie point symmetries and demonstrate the effectiveness of this symmetry regularization on generalization capabilities.

2. Background

PDEs are used to describe the dynamics of various physical systems mathematically. In a PDE, the evolution of a function that involves several variables is described in terms of local updates expressed by partial derivatives. We consider PDEs of the following general form:

$$\begin{aligned} \Delta = \mathbf{u}_t + \mathcal{D}_{\mathbf{x}}[\mathbf{u}] &= 0, && t \in [0,T], \mathbf{x} \in \Omega \\ \mathbf{u}(0,\mathbf{x}) &= f(\mathbf{x}), && \mathbf{x} \in \Omega \\ \mathbf{u}(t,\mathbf{x}) &= g(t,\mathbf{x}), && \mathbf{x} \in \partial\Omega, t \in [0,T] \end{aligned} \quad (1)$$

where $u(t,\mathbf{x}) \in \mathbb{R}^{d_u}$ is the solution to the PDE that we seek, t denotes the time, and $\mathbf{x}$ a vector of spatial coordinates.

[1]School of Computer Science, McGill University [2]Mila - Quebec Artficial Intelligence Institute [3]Universite de Montréal, Montreal, Quebec, Canada [4]CIELA Institute, Montreal, Quebec, Canada [5]Microsoft Research AI4Science [6]University of Amsterdam, Amsterdam, Netherlands. Correspondence to: Tara Akhound-Sadegh <tara.akhoundsadegh@mila.quebec>.

Presented at the 2^{nd} Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

$\Omega \subset \mathbb{R}^D$ is the domain and $\mathcal{D}_{\mathbf{x}}[.]$ is a non-linear *differential operator*. $f(\mathbf{x})$ is known as the initial condition function and $g(t, \mathbf{x})$ describes the boundary conditions.

For many systems, obtaining an analytical solution to the PDE is impossible; hence, numerical methods are traditionally used to obtain approximate solutions. Numerical solvers, such as finite element methods (FEM) or finite difference methods (FDM) rely on discretizing the space $T \times \Omega$ (Quarteroni, 2009). The topology of the space has to be taken into account when constructing the mesh, and the resolution of the discretization will affect the accuracy of the predicted solutions. Additionally, these solvers are often computationally expensive, especially for complex dynamics, and each time the initial or boundary conditions of the PDE change the solver must be rerun. These considerations and constraints make designing numerical solvers difficult, and scientists often need to handcraft a specific solver for an application (Quarteroni, 2009).

Two main approaches exist to use deep learning to solve PDEs: Neural Operator Methods (NO), which are trained with numerical PDE solutions as targets, and Physics-Informed Neural Networks (PINNs), which assume knowledge of the governing equation of the PDE. For examples of NO methods see (Lu et al., 2021; Kovachki et al., 2023; Li et al., 2020; 2021; Bhattacharya et al., 2021; Patel et al., 2021; Bar-Sinai et al., 2019; Brandstetter et al., 2022b; Sanchez-Gonzalez et al., 2020). This work presents an extension to PINNs, which we review in detail in Section 3.1.

2.1. Symmetries of a Partial Differential Equation

The exposition in this section mainly follows (Olver, 1986), and we refer the reader to this original text for more details on the topic.

Symmetries are transformations of the object that leave an aspect of it invariant. In the context of PDE, these are transformations that map a solution of the PDE to another solution. For example, in Section 1, the solutions of a simple harmonic oscillator can be obtained via symmetry transformations of a given solution.

Consider the PDE Δ, involving p independent variables $\mathbf{x} = (x_1, \dots, x_p) \in X$ and q dependent variables (solutions) $\mathbf{u} = (u_1, \dots, u_q) \in U$. The symmetry group $\mathcal{G}$, of Δ, is the local group of transformations on an open subset of the space of dependent and independent variables, $M \subset X \times U$, which transforms solutions Δ to other solutions of Δ.

Prolongations. To formalize this abstract definition of PDE symmetries, Lie proposed viewing Δ as a concrete geometric object and introduced the concept of *prolongation* (Olver, 1986).

Definition 1. *The **n-th order prolongation** (or n-th order* jet space*) of $X \times U$ is denoted as $X \times U^{(n)} = X \times U_1 \times \dots \times U_n$, whose coordinates represent the independent and dependent variables as well as all the partial derivatives of the dependent variables up to order n.*

Equivalently we have the notion of *prolongation* of $\mathbf{u}$ as $\mathbf{u}^{(n)} = (\mathbf{u}_{\mathbf{x}}, \mathbf{u}_{\mathbf{xx}}, \dots, \mathbf{u}_{n\mathbf{x}})$ where $\mathbf{u}_{i\mathbf{x}}$ is all the unique i^{th} derivatives of u, for $i = 1, \dots, n$. For example, if $\mathbf{x} = (x, y)$, then $\mathbf{u}_{\mathbf{xx}} = (\partial_{xx}\mathbf{u}, \partial_{xy}\mathbf{u}, \partial_{yy}\mathbf{u})$. Using this notion of prolongation, we can represent a PDE as an algebraic equation, $\Delta(\mathbf{x}, \mathbf{u}^{(n)}) = 0$, where Δ is the map that determines the PDE, *i.e.*, $\Delta : X \times U^{(n)} \to \mathbb{R}$. In other words, the PDE tells us where the map Δ vanishes on $X \times U^{(n)}$.

For example, for the one-dimensional heat equation describing the heat conduction in a one-dimensional rod, with viscosity ν, we have:

$$\Delta((x,t), \mathbf{u}^{(2)}) = u_t - \nu u_{xx} \tag{2}$$

Prolongations of the Infinitesimal Generators. Let $\mathbf{v}$ be the vector field on the subspace $M \subset X \times U$ with corresponding one-parameter subgroup $\exp(\epsilon \mathbf{v})$ (see Appendix B for the definition of one-parameter subgroups). In other words, the vector field $\mathbf{v}$ is the *infinitesimal generator* of the one-parameter subgroup. Intuitively, this vector field describes the infinitesimal transformations of the group to the independent and dependent variables, and we can write it as:

$$\mathbf{v} = \sum_{i=1}^{p} \xi_i(\mathbf{x}, \mathbf{u}) \frac{\partial}{\partial x^i} + \sum_{\alpha=1}^{q} \phi_\alpha(\mathbf{x}, \mathbf{u}) \frac{\partial}{\partial u^\alpha} \tag{3}$$

where $\xi^i(\mathbf{x}, \mathbf{u})$ and $\phi_\alpha(\mathbf{x}, \mathbf{u})$ are coordinate-dependent coefficients. To study how symmetries transform one solution to another, we need to know how they transform the partial derivatives and, therefore, the jet space.

A symmetry transformation of the independent ($\mathbf{x}$) and dependent ($\mathbf{u}$) variables will also induce transformations in the partial derivatives $\mathbf{u}_{\mathbf{x}}, \mathbf{u}_{\mathbf{xx}}, \dots$. The *prolongation of the infinitesimal generator*, $\mathrm{pr}^{(n)}\mathbf{v}$ is a generalization of the generator $\mathbf{v}$ which describes this induced transformations. This prolongation is defined on the jet-space $X \times U^{(n)}$ and is given by:

$$\mathrm{pr}^{(n)}\mathbf{v} = \sum_{i=1}^{p} \xi_i(\mathbf{x}, \mathbf{u}) \frac{\partial}{\partial x^i} + \sum_{\alpha=1}^{q} \sum_{J} \phi_\alpha^{(J)}(\mathbf{x}, \mathbf{u}) \frac{\partial}{\partial u_J^\alpha} \tag{4}$$

where we have used the notation $J = (i_1, \dots, i_k)$ for the multi-indices, with $0 \le i_k \le p$ and $0 \le k \le n$ and $\mathbf{u}_J^\alpha = \frac{\partial^k \mathbf{u}^\alpha}{\partial x^{i_1} \dots \partial x^{i_k}}$ Calculating $\phi_\alpha^{(J)}$, the coefficients of $\partial_{\mathbf{u}_J^\alpha}$, can be done using the prolongation formula, given in Eq. (12), which involves the total derivative operator D. See Appendix A for an example of this calculation.

The upshot is that we can mechanically calculate the prolonged vector field using partial derivatives of $\mathbf{u}$, which are produced by automatic differentiation. In practice, prolonged vector fields are implemented as vector-valued functions (or functionals) of $\mathbf{x}$ and $\mathbf{u}$. The implementation of this process is generic and can be applied to any PDE.

Lie Point Symmetry.

Definition 2. *For symmetry group $\mathcal{G}$ acting on M, the* ***prolongation of action of*** *$\mathcal{G}$ on the open subset $M \subset X \times U$ is the induced action on $M^{(n)} = M \times U^{(n)}$ which transforms derivatives of $\mathbf{u} = f(\mathbf{x})$ into corresponding derivatives of $\mathbf{u}' = f'(\mathbf{x}')$. We can write this as:* $\mathrm{pr}^{(n)}g \cdot (\mathbf{x}, \mathbf{u}^{(n)}) = (g \cdot \mathbf{x}, \mathrm{pr}^{(n)}g \cdot \mathbf{u}^{(n)})$

Using this definition, we provide a new criterion for $\mathcal{G}$ being a symmetry group of Δ, under a mild assumption on the PDE.[1]

Theorem 2.1. *$\mathcal{G}$ is the symmetry group of the n-th order PDE $\Delta(\mathbf{x}, \mathbf{u}^n)$, if $\mathcal{G}$ acts on M, and its prolongation leaves the solution set $\mathcal{S}_\Delta$ invariant: $\mathrm{pr}^{(n)}g \cdot (\mathbf{x}, \mathbf{u}^{(n)}) \in \mathcal{S}_\Delta, \forall g \in \mathcal{G}$. (for a formal definition of the solution set, refer to Appendix B, Eq. (11)).*

Finally, we can express the symmetry condition in terms of the infinitesimal generators $\mathbf{v}$ of $\mathcal{G}$ and refer to the Appendix A for an example of applying the infinitesimal criterion to the heat equation.

Theorem 2.2 (Infinitesimal Criterion)**.** *$\mathcal{G}$ is a symmetry group of the PDE $\Delta(\mathbf{x}, \mathbf{u}^n)$ if for every infinitesimal generator $\mathbf{v}$ of $\mathcal{G}$, we have that $\mathrm{pr}^{(n)}\mathbf{v}[\Delta] = 0$ when $\Delta = 0$*

3. Methods

3.1. Introduction to PINNs

NO methods are trained with a supervised loss, which is infeasible in many situations. Therefore, as an alternative, PINNs have been proposed (Raissi et al., 2019). In this framework, the *neural surrogate model* for the PDE solution is trained directly with the PDE itself. The simplicity of the PINNs idea has made it the subject of many follow-up improvements; see (Krishnapriyan et al., 2021; Wang et al., 2020b; 2022)

In PINNs, the PDE solution $u(t, \mathbf{x})$ of Eq. (1) is a neural network $u_\theta(t, \mathbf{x})$ with parameters θ. The loss function is:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{data-fit}} \tag{5}$$

The first term is the *physics-informed* objective, ensuring that the function learned by the network satisfies Eq. (1):

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_r} \sum_{i=1}^{N_l} \| \frac{\partial}{\partial t} u_\theta(t_i, \mathbf{x}_i) + \mathcal{D}_{\mathbf{x}}[u_\theta(t_i, \mathbf{x_i})\|_2^2 \tag{6}$$

where the derivatives of the solution network u_θ are calculated using automatic differentiation (Raissi et al., 2019), and the penalty is then imposed on a finite set of points $(t, \mathbf{x})_{1:N_r}$, sampled from inside the domain $[0, T] \times \Omega$.

The second term in Eq. (5), is a supervised loss, ensuring that the function learned by the neural network satisfies the initial and boundary conditions of the problem:

$$\begin{aligned}\mathcal{L}_{\text{data-fit}} = &\frac{1}{N_0} \sum_{i=1}^{N_0} \| u_\theta(0, \mathbf{x}_i^0) - f(x_i^0) \|_2^2 \\ &+ \frac{1}{N_b} \sum_{i=1}^{N_b} \| u_\theta(t_i^b, \mathbf{x}_i^b) - g(t_i^b, x_i^b) \|_2^2\end{aligned} \tag{7}$$

where $(\mathbf{x}^0)_{1:N_0}$ are N_0 samples from Ω at which the initial condition function f is sampled. $(t^b, \mathbf{x}^b)_{1:N_b}$ are N_b points sampled on the boundary (from $[0, T] \times \partial\Omega$).

3.2. Solving PDEs with Different Initial/Boundary Conditions with PINNs

Wang et al. (2021b) combines the NO method approach introduced in (Lu et al., 2021) with the PINN loss. We will also use their framework to examine the effect of enforcing the symmetry condition of the PDE on the model. The model consists of two neural networks: e_{θ_1} embeds the initial condition function, and g_{θ_2} embeds the independent variables, $[t, \mathbf{x}] \in \mathbb{R}^p$.

In particular, to embed the initial condition function $f(\mathbf{x}) = \mathbf{u}(0, \mathbf{x}) \in \mathbb{R}^p$, it is sampled at fixed points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and the concatenated values are fed to f_{θ_1}. Using the notation $\theta = (\theta_1, \theta_2)$, the final prediction is the inner product of these embedding vectors:

$$\mathcal{O}_\theta(f)(\mathbf{x}, t) = e_{\theta_1}^\top \big(\mathbf{u}(0, \mathbf{x}_1), \ldots, \mathbf{u}(0, \mathbf{x}_n)\big)\, g_{\theta_2}(\mathbf{x}, t) \tag{8}$$

3.3. Imposing the Symmetry Criterion

To further inform PINNs about the symmetries of the PDE, we use an additional loss term $\mathcal{L}_{\text{sym}}$.

The infinitesimal criterion of Theorem 2.2 requires that by acting on a solution $(\mathbf{x}, \mathbf{u}^n)$ in the jet space using the prolonged infinitesimal generator $\mathrm{pr}^{(n)}\mathbf{v}$, the PDE should remain satisfied. Our symmetry loss encourages the orthogonality of $\mathrm{pr}^{(n)}\mathbf{v}$ and the gradient of Δ.

Assume that the Lie algebra of the symmetry group of the n-th order PDE, Δ, is spanned by K independent vector fields, $\{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$, where each $\mathbf{v}_k$ is defined as in Eq. (3). For each $\mathbf{v}_k$, we can obtain their prolongations using automatic differentiation and create a vector of the corresponding coefficients, which we call $\mathrm{coef}(\mathrm{pr}^{(n)})$.[2] We also use the notation J_Δ for the gradient of Δ wrt all independent and dependent variables: $J_\Delta = \left(\frac{\partial \Delta}{\partial x^i}, \frac{\partial \Delta}{\partial \mathbf{u}_J^\alpha}\right)$.

[1]This assumption is that Δ is of maximal rank. We refer to Olver (1986) for the definition of this condition. However, we note that this assumption does not pose a restriction since for any PDE not satisfying this condition, it is possible to find an equivalent PDE which does.

[2]Using coef in the equation above is to differentiate the abstract definition of Eq. (3) and a vector of its coefficients.

The symmetry loss encourages the orthogonality of the K prolonged vector fields and the gradient vector:

$$\mathcal{L}_{\text{sym}} = \sum_{k=1}^{K} J_{\Delta}^{\top} \text{coef}(\text{pr}^{(n)} \mathbf{v}_k) \tag{9}$$

Therefore, the total loss consists of the PINN loss, Eq. (5), and the symmetry loss: $\mathcal{L} = \alpha \mathcal{L}_{\text{PDE}} + \beta \mathcal{L}_{\text{data-fit}} + \gamma \mathcal{L}_{\text{sym}}$, for α, β and γ hyperparameters. However, as we see through examples, one or more symmetries of a PDE often simplify to a constant times the $\mathcal{L}_{PDE}$, removing the need to separate treatment of the PDE loss. Algorithm 1 in Appendix D summarizes our training algorithm.

4. Experiment: Heat Equation

We study the effectiveness of imposing the symmetry constraint on the heat equation, described in Eq. (2). Additional results for Burgers' equation are included in Appendix E.

Symmetries. The following 6-dimensional lie algebra spans the symmetry group of the heat equation:

$$\begin{aligned} \mathbf{v}_1 &= \partial_x & \mathbf{v}_6 &= 4\nu t x \partial_x - 4\nu t^2 \partial_t - (x^2 + 2\nu t) u \partial_u \\ \mathbf{v}_2 &= \partial_t & \mathbf{v}_4 &= x\partial_x + 2t\partial_t \\ \mathbf{v}_3 &= \partial_u & \mathbf{v}_6 &= 4\nu t x \partial_x - 4\nu t^2 \partial_t - (x^2 + 2\nu t) u \partial_u \end{aligned} \tag{10}$$

For example, the infinitesimal generator v_1 corresponds to space translation, and v_5 to Galilean boost. We refer the reader to (Olver, 1986) for the derivation of these generators and their corresponding one-parameter groups.

Training and Experiments. We want to confirm that the symmetry loss helps improve the model's prediction capability, especially in a low-data regime. Therefore, we train the model with and without symmetry and evaluate the predictions on the test dataset as we increase the number of samples inside the domain, N_r. To illustrate the effectiveness of the symmetries in a low-data regime, we use $N_f = 100$ different initial conditions and test the performance as we increase N_r from 500 to 2000 and 10000. We refer to Appendix C for details on the data generation process and to Appendix D for the architectures and hyperparameters.

Results. Table 1 compares the performance of the two models on the test dataset of unobserved initial conditions as N_r increases. When trained with few samples, the model trained with the symmetry loss performs significantly better than the baseline model. Fig. 2 also illustrates this point as it shows the performance of both models on a single instance from the test dataset. We note that the improvement in prediction results in the model trained with $\mathcal{L}_{\text{sym}}$ is especially significant at larger values of time, t.
Importantly we highlight that by using the infinitesimal criterion for enforcing symmetries, not all symmetries will help improve the training. There are instances when the gradient of Δ along the vector field is trivially zero, and in others, we obtain $c\Delta$ for c a constant. In the case of the heat equation, only v_5 and v_6 provide training signals different from the $\mathcal{L}_{\text{PDE}}$. This means that in our experiments, we can eliminate the PDE loss and use the symmetry loss instead.

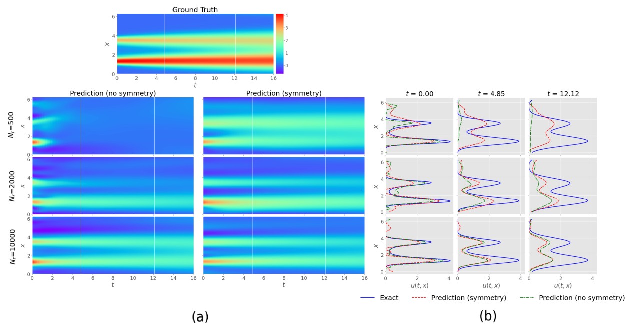

Figure 2. The effect of training the PDE solver for the heat equation with and without the symmetry loss for one of the PDEs in the test dataset. (a) shows the ground truth solution and the predictions of the two models as the number of samples inside the domain increases from 500 to 2000 and 1000.(b) shows the corresponding predictions and the ground truth solution at different time slices.

Table 1. The average test set mean-squared error for the Heat equation.

Number of Points (N_r)	No Symmetry	Symmetry
500	1.12 ± 0.58	$\mathbf{0.30 \pm 0.15}$
2000	0.36 ± 0.19	$\mathbf{0.24 \pm 0.14}$
1000	0.22 ± 0.14	$\mathbf{0.21 \pm 0.13}$

Conclusion and Limitations

Our work presents the foundations for leveraging Lie point symmetry in a large family of Neural PDE solvers that do not require access to simulated data. In this work, we show that local symmetry constraints can improve PDE solutions found using PINN models.

Our proposed method has some limitations: 1) while the Lie point symmetries of important PDEs are well-known, in general, one needs to analytically derive them to use our approach; 2) as we mentioned in Section 4, not all symmetries of the equation will necessarily be useful for constraining the PINN. Fortunately, the usefulness of symmetries is obvious from the corresponding infinitesimal generator, and one could limit the symmetry loss to useful symmetries; 3) while symmetries can significantly improve performance, based on our observations (see Table 1) one could achieve a similar effect with PINN by increasing the sample size. These limitations motivate our future direction, which builds on our current understanding, to impose symmetry constraints through equivariant architectures.

References

Bar-Sinai, Y., Hoyer, S., Hickey, J., and Brenner, M. P. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. doi: 10.1073/pnas.1814058116.

Bhattacharya, K., Hosseini, B., Kovachki, N. B., and Stuart, A. M. Model reduction and neural networks for parametric pdes, 2021.

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*, volume 162, pp. 2241–2256. PMLR, 2022a.

Brandstetter, J., Worrall, D. E., and Welling, M. Message passing neural PDE solvers. In *The 10th International Conference on Learning Representations*. OpenReview.net, 2022b.

Kashinath, K., Mustafa, M., Albert, A., Wu, J.-L., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., Manepalli, A., Chirila, D., Yu, R., Walters, R., White, B., Xiao, H., Tchelepi, H. A., Marcus, P., Anandkumar, A., Hassanzadeh, P., and Prabhat. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200093, February 2021.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning – accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021. doi: 10.1073/pnas.2101784118.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces, 2023.

Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pp. 26548–26560, 2021.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. 2020.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021.

Olver, P. J. *Applications of Lie Groups to Differential Equations*. Springer, 1986.

Patel, R. G., Trask, N. A., Wood, M. A., and Cyr, E. C. A physics-informed operator regression framework for extracting data-driven continuum models. *Computer Methods in Applied Mechanics and Engineering*, 373:113500, 2021. doi: 10.1016/j.cma.2020.113500.

Quarteroni, A. *Numerical Models for Differential Problems*, volume 2. Springer, 2009.

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.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. W. Learning to simulate complex physics with graph networks, 2020.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. Towards physics-informed deep learning for turbulent flow prediction, 2020a.

Wang, R., Walters, R., and Yu, R. Incorporating symmetry into deep dynamics models for improved generalization, 2021a.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient pathologies in physics-informed neural networks, 2020b.

Wang, S., Wang, H., and Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40), 2021b.

Wang, S., Sankaran, S., and Perdikaris, P. Respecting causality is all you need for training physics-informed neural networks, 2022.