The Mathematics of Physics-Informed Neural Networks vs. Physics-Informed Neural Operators

Miquel Noguer i Alonso

Artificial Intelligence Finance Institute

June 16, 2025

Abstract

Physics-informed machine learning has emerged as a transformative approach for solving partial differential equations (PDEs) by incorporating physical laws into neural network architectures. Two prominent paradigms have emerged: Physics-Informed Neural Networks (PINNs) and Physics-Informed Neural Operators (PINOs). While PINNs learn specific solutions to PDEs by embedding physical constraints into the loss function, PINOs learn operators that map between function spaces, enabling parametric studies and faster inference. This paper provides a comprehensive mathematical and computational comparison of these approaches, analyzing their theoretical foundations, approximation capabilities, computational complexity, and practical applications. We discuss the trade-offs between solution-specific accuracy (PINNs) and operator-level generalization (PINOs), providing guidance for practitioners on method selection based on problem characteristics. Our analysis reveals that PINNs excel in complex geometries and inverse problems, while PINOs demonstrate superior performance in parametric studies and real-time applications.

1 Introduction

The intersection of deep learning and scientific computing has given rise to physics-informed machine learning, a paradigm that incorporates physical laws and constraints into neural network architectures [Karniadakis et al., 2021]. Two major approaches have emerged as leading methodologies: Physics-Informed Neural Networks (PINNs) and Physics-Informed Neural Operators (PINOs).

PINNs, introduced by Raissi et al. [2019], revolutionized the field by demonstrating how neural networks could solve forward and inverse problems involving nonlinear PDEs by embedding physical laws directly into the loss function. This approach enables the network to learn solutions that satisfy both the governing equations and boundary/initial conditions simultaneously.

In contrast, PINOs represent a paradigm shift from learning specific solutions to learning operators that map between function spaces [Li et al., 2020, Wang et al., 2021]. This operator learning approach, built upon the foundation of neural operators [Li et al., 2021], enables the trained model to generalize across different parameter configurations without retraining.

The fundamental question addressed in this paper is: *When should practitioners choose PINNs versus PINOs, and what are the mathematical and computational trade-offs inherent in each approach?* We provide a comprehensive analysis of both methods, examining their theoretical foundations, approximation capabilities, computational complexity, and practical limitations.

2 Mathematical Foundations

2.1 Physics-Informed Neural Networks (PINNs)

PINNs address the problem of solving PDEs of the general form:

$$\mathcal{F}[u](x,t) = f(x,t), \quad (x,t) \in \Omega \times [0,T] \tag{1}$$

subject to boundary conditions:

$$\mathcal{B}[u](x,t) = g(x,t), \quad (x,t) \in \partial\Omega \times [0,T] \tag{2}$$

and initial conditions:

$$u(x,0) = u_0(x), \quad x \in \Omega \tag{3}$$

where $\mathcal{F}$ is a differential operator, $\Omega \subset \mathbb{R}^d$ is the spatial domain, and $\mathcal{B}$ represents boundary operators.

The PINN approximation $u_\theta(x,t)$, parameterized by neural network weights θ, is trained by minimizing the composite loss function:

$$\mathcal{L}(\theta) = \mathcal{L}_{PDE}(\theta) + \lambda_{BC}\mathcal{L}_{BC}(\theta) + \lambda_{IC}\mathcal{L}_{IC}(\theta) \tag{4}$$

where:

$$\mathcal{L}_{PDE}(\theta) = \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} |\mathcal{F}[u_\theta](x_i,t_i) - f(x_i,t_i)|^2 \tag{5}$$

$$\mathcal{L}_{BC}(\theta) = \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} |\mathcal{B}[u_\theta](x_i,t_i) - g(x_i,t_i)|^2 \tag{6}$$

$$\mathcal{L}_{IC}(\theta) = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} |u_\theta(x_i,0) - u_0(x_i)|^2 \tag{7}$$

The key innovation lies in using automatic differentiation to compute the required derivatives $\frac{\partial u_\theta}{\partial x}$, $\frac{\partial u_\theta}{\partial t}$, etc., enabling exact enforcement of differential constraints [Raissi et al., 2017].

2.2 Physics-Informed Neural Operators (PINOs)

PINOs learn operators $\mathcal{G}_\theta$ that map input functions to output functions:

$$\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U} \tag{8}$$

where $\mathcal{A}$ and $\mathcal{U}$ are function spaces. For a parametric PDE:

$$\mathcal{F}[u;a](x,t) = 0 \tag{9}$$

the operator maps parameter functions $a(x) \in \mathcal{A}$ to solution functions $u(x,t) \in \mathcal{U}$.

The PINO loss function combines data-driven and physics-informed terms:

$$\mathcal{L}(\theta) = \mathcal{L}_{data}(\theta) + \lambda_{physics}\mathcal{L}_{physics}(\theta) \tag{10}$$

where:

$$\mathcal{L}_{data}(\theta) = \mathbb{E}_{a\sim\mu}\|[\mathcal{G}_\theta(a)](x) - u(x)\|^2 \tag{11}$$

$$\mathcal{L}_{physics}(\theta) = \mathbb{E}_{a\sim\mu}\|\mathcal{F}[[\mathcal{G}_\theta(a)];a]\|^2 \tag{12}$$

The expectation is taken over a distribution μ of input parameters, enabling the operator to generalize across the parameter space [Li et al., 2023].

2.3 Neural Operator Architectures

PINOs typically employ Fourier Neural Operators (FNOs) [Li et al., 2020] or DeepONets [Lu et al., 2021]. The FNO architecture performs computations in the Fourier domain:

$$(\mathcal{K}(w) \cdot v)(x) = \mathcal{F}^{-1}(R_w \cdot (\mathcal{F}v))(x) \tag{13}$$

where $\mathcal{F}$ denotes the Fourier transform, R_w is a learnable linear transformation, and w represents the network parameters.

3 Theoretical Analysis

3.1 Mathematical Foundations and Function Spaces

3.1.1 Sobolev Space Framework

Both PINNs and PINOs operate within Sobolev space frameworks. For PINNs solving PDEs on domain $\Omega \subset \mathbb{R}^d$, the solution u typically belongs to appropriate Sobolev spaces $H^k(\Omega)$ with norm:

$$\|u\|^2_{H^k(\Omega)} = \sum_{|\alpha| \leq k} \int_\Omega |D^\alpha u(x)|^2 dx \tag{14}$$

where D^α denotes weak derivatives of order $|\alpha|$.

For PINOs, we consider operators between function spaces. Let $\mathcal{A} = L^2(\mathcal{D}; \mathbb{R}^{d_a})$ and $\mathcal{U} = H^s(\mathcal{D}; \mathbb{R}^{d_u})$ where $\mathcal{D} \subset \mathbb{R}^d$ is the spatial domain. The operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ maps input functions to solution functions in appropriate Sobolev spaces.

3.1.2 Variational Formulation

PINNs can be viewed through the lens of variational methods. Consider the weak formulation:

$$\text{Find } u \in V \text{ such that } a(u, v) = F(v) \quad \forall v \in V \tag{15}$$

where V is an appropriate function space, $a(\cdot, \cdot)$ is a bilinear form, and F is a linear functional.

The PINN approximation u_θ minimizes:

$$J(u_\theta) = \|R[u_\theta]\|^2_{L^2} + \lambda_{BC}\|B[u_\theta]\|^2_{L^2(\partial\Omega)} \tag{16}$$

where $R[u_\theta]$ is the PDE residual and $B[u_\theta]$ represents boundary condition violations.

3.2 Approximation Theory

3.2.1 Universal Approximation for PINNs

Theorem 1 (Universal Approximation for PINNs): Let $f : \mathbb{R}^d \to \mathbb{R}$ be continuous and bounded. For any $\epsilon > 0$, there exists a feed-forward neural network f_θ with ReLU activations such that:

$$\sup_{x \in K} |f(x) - f_\theta(x)| < \epsilon \tag{17}$$

for any compact set $K \subset \mathbb{R}^d$.

However, approximation in function space norms is more subtle. The following theorem provides rates:

Theorem 2 (Sobolev Approximation Rates): Let $u \in H^{k+1}(\Omega)$ be the true solution. If the neural network u_θ has width W and depth L, then:

$$\|u - u_\theta\|_{H^s(\Omega)} \leq CW^{-k/d} + \mathcal{O}(h^{k+1-s}) \tag{18}$$

where h is the mesh parameter for collocation points, $s \leq k$, and C depends on $\|u\|_{H^{k+1}}$.

3.2.2 Operator Approximation Theory

For neural operators, we have stronger theoretical guarantees:

Theorem 3 (Operator Universal Approximation): Let $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ be a bounded operator between separable Banach spaces. Then for any $\epsilon > 0$, there exists a neural operator $\mathcal{G}_\theta$ such that:

$$\|\mathcal{G} - \mathcal{G}_\theta\|_{\mathcal{L}(\mathcal{A},\mathcal{U})} < \epsilon \tag{19}$$

Corollary: For Fourier Neural Operators with sufficient modes N and depth L:

$$\|\mathcal{G} - \mathcal{G}_\theta\|_{\mathcal{L}(\mathcal{A},\mathcal{U})} \leq CN^{-r} + \tilde{C}L^{-s} \tag{20}$$

where $r, s > 0$ depend on the smoothness of $\mathcal{G}$.

3.3 Convergence Analysis

3.3.1 PINNs Convergence Theory

Strong Convergence Results: Under smoothness assumptions on the true solution u:

$$\lim_{N\to\infty} \mathbb{E}[\|u_N - u\|^2_{H^1(\Omega)}] = 0 \tag{21}$$

where u_N is the PINN solution with N collocation points.

Convergence Rates: For second-order elliptic PDEs with smooth coefficients:

$$\mathbb{E}[\|u_N - u\|^2_{L^2(\Omega)}] \leq CN^{-2k/d} \tag{22}$$

where k is the regularity index of u and d is the spatial dimension.

A Priori Error Estimates: Let u_θ^{opt} be the best approximation in the neural network space. Then:

$$\|u - u_\theta\|_{H^1} \leq \|u - u_\theta^{opt}\|_{H^1} + \|\mathcal{L}^{-1}\| \cdot \|R[u_\theta^{opt}]\|_{L^2} \tag{23}$$

where $\mathcal{L}^{-1}$ is the solution operator and $R[\cdot]$ is the residual operator.

3.3.2 PINOs Convergence Analysis

Statistical Learning Theory: For PINOs trained on M parameter samples:

$$\mathbb{E}[\|\mathcal{G} - \mathcal{G}_M\|^2_{\mathcal{L}(\mathcal{A},\mathcal{U})}] \leq C_1 M^{-\alpha} + C_2 \mathcal{R}_M \tag{24}$$

where $\mathcal{R}_M$ is the empirical Rademacher complexity and $\alpha > 0$ depends on the parameter distribution.

Operator Learning Rates: For operators with finite rank representations:

$$\|\mathcal{G} - \mathcal{G}_\theta\|_{\text{op}} \leq \sqrt{\frac{\text{rank}(\mathcal{G})\log(d)}{M}} + \text{bias}(\theta) \tag{25}$$

3.4 Optimization Landscape Analysis

3.4.1 Loss Surface Geometry for PINNs

The PINN loss landscape exhibits complex geometry due to the interaction between data fidelity and physics constraints. Define the total loss:

$$\mathcal{L}(\theta) = \mathcal{L}_{data}(\theta) + \lambda \mathcal{L}_{physics}(\theta) \tag{26}$$

Hessian Analysis: The Hessian of the physics loss has the structure:

$$H_{physics} = \nabla^2 \mathcal{L}_{physics} = \sum_{i=1}^{N} \nabla_\theta R[u_\theta](x_i) \nabla_\theta R[u_\theta](x_i)^T + \text{second-order terms} \tag{27}$$

Conditioning: The condition number of the Hessian scales as:

$$\kappa(H) \leq C \cdot \max\left(\frac{\lambda_{max}(\mathcal{A})}{\lambda_{min}(\mathcal{A})}, \frac{1}{\lambda}\right) \tag{28}$$

where $\mathcal{A}$ is the differential operator and λ is the physics weighting parameter.

3.4.2 Spectral Bias in PINNs

Neural networks exhibit spectral bias, preferring low-frequency components. For PINNs:

Spectral Decomposition: If $u(x) = \sum_k a_k \phi_k(x)$ where $\{\phi_k\}$ are eigenfunctions, then:

$$\mathbb{E}[|a_k^{(N)} - a_k|^2] \propto \lambda_k^{-2\beta} \tag{29}$$

where λ_k are eigenvalues and $\beta > 0$ characterizes the spectral bias.

Mitigation Strategies: Frequency-enhanced training modifies the loss:

$$\tilde{\mathcal{L}}(\theta) = \sum_k w_k \lambda_k^\gamma |\langle R[u_\theta], \phi_k \rangle|^2 \tag{30}$$

where w_k and γ are chosen to balance frequency content.

3.5 Generalization Theory

3.5.1 Rademacher Complexity for PINNs

Definition: The Rademacher complexity for the PINN function class $\mathcal{F}_\theta$ is:

$$\mathcal{R}_N(\mathcal{F}_\theta) = \mathbb{E}_{\sigma, X}\left[\sup_{f \in \mathcal{F}_\theta} \frac{1}{N} \sum_{i=1}^{N} \sigma_i f(x_i)\right] \tag{31}$$

where σ_i are independent Rademacher variables.

Bounds: For neural networks with Lipschitz constant L and depth D:

$$\mathcal{R}_N(\mathcal{F}_\theta) \leq \frac{CL\sqrt{D \log(W))}}{\sqrt{N}} \tag{32}$$

Generalization Bound: With probability $1 - \delta$:

$$|\mathcal{L}_{test} - \mathcal{L}_{train}| \leq 2\mathcal{R}_N(\mathcal{F}_\theta) + \sqrt{\frac{\log(1/\delta)}{2N}} \tag{33}$$

3.5.2 Physics-Informed Generalization

The physics constraints provide implicit regularization:

Physics Regularization Effect: The effective capacity of the function class is reduced by physics constraints:

$$\mathcal{R}_{eff}(\mathcal{F}_\theta^{physics}) \leq \mathcal{R}(\mathcal{F}_\theta) \cdot \sqrt{1 - \rho^2} \tag{34}$$

where ρ measures the correlation between data and physics terms.

3.6 Information-Theoretic Analysis

3.6.1 Information Content of Physics Constraints

Mutual Information: The physics constraints provide information $I(U; \Phi)$ where U is the solution and Φ represents

3.6.2 PINNs Complexity

For PINNs, the computational complexity per training iteration is $\mathcal{O}(N \cdot D \cdot W)$ where:

- N is the number of collocation points
- D is the network depth
- W is the network width

The forward pass requires evaluating the network and computing derivatives via automatic differentiation. The derivative computation adds a multiplicative factor related to the order of derivatives required by the PDE.

3.6.3 PINOs Complexity

PINOs have different complexity characteristics:

- Training complexity: $\mathcal{O}(M \cdot B \cdot C_{op})$ where M is the number of parameter samples, B is the batch size, and C_{op} is the operator evaluation cost
- Inference complexity: $\mathcal{O}(C_{op})$ independent of the specific parameter values

For FNOs, $C_{op} = \mathcal{O}(N \log N)$ due to FFT operations, making them highly efficient for inference [Li et al., 2020].

4 Comparative Analysis

4.1 Problem Suitability

Table 1: Comparison of PINNs vs PINOs across different problem characteristics

Characteristic	**PINNs**	**PINOs**
Complex geometries	Excellent	Limited
Irregular boundaries	Excellent	Moderate
Parametric studies	Poor	Excellent
Real-time inference	Poor	Excellent
Inverse problems	Excellent	Moderate
Limited data scenarios	Excellent	Poor
Multi-scale problems	Good	Excellent
High-dimensional problems	Moderate	Good

4.2 Accuracy vs. Efficiency Trade-offs

PINNs typically achieve higher accuracy for specific problem instances due to their ability to exactly enforce boundary conditions and physical constraints. However, this comes at the cost of requiring retraining for each new parameter configuration.

PINOs sacrifice some accuracy for individual problems but gain significant efficiency in parametric studies. Goswami et al. [2022] demonstrated that PINOs can achieve comparable accuracy to PINNs while providing orders of magnitude speedup for parametric sweeps.

4.3 Data Requirements

PINNs can operate in purely physics-informed modes with minimal or no observational data, relying primarily on the governing equations. This makes them particularly suitable for scenarios where experimental data is scarce or expensive to obtain.

PINOs require training data covering the parameter space of interest, making them more data-hungry initially but more efficient for subsequent queries within the trained parameter range.

5 Applications and Case Studies

5.1 Fluid Dynamics

Jin et al. [2021] developed NSFnets using PINNs for incompressible Navier-Stokes equations, demonstrating excellent performance in complex flow scenarios. The ability to handle irregular geometries and incorporate boundary conditions naturally makes PINNs well-suited for computational fluid dynamics applications.

In contrast, Li et al. [2020] showed that FNO-based approaches excel in turbulence modeling and parametric studies of fluid systems, where multiple simulations with varying parameters are required.

5.2 Heat Transfer and Thermal Management

Howard et al. [2022] applied multifidelity PINOs to nanoscale heat transport problems, demonstrating the ability to perform rapid inverse design across parameter spaces. This application highlights the strength of PINOs in engineering design optimization.

5.3 Uncertainty Quantification

Yang et al. [2021] introduced Bayesian PINNs (B-PINNs) for forward and inverse problems with noisy data, extending the PINN framework to handle uncertainty. The physics-informed constraints help regularize the uncertainty estimates, providing more reliable predictions than purely data-driven approaches.

6 Challenges and Limitations

6.1 PINNs Limitations

6.1.1 Training Difficulties

PINNs can suffer from training pathologies, including:

- Loss balancing issues between different loss components
- Spectral bias favoring low-frequency solutions

- Convergence difficulties for stiff PDEs

Krishnapriyan et al. [2021] identified several failure modes and proposed diagnostic tools for detecting and mitigating these issues.

6.1.2 Computational Scalability

For large-scale problems, PINNs can become computationally expensive due to the need for numerous collocation points and the cost of automatic differentiation for higher-order derivatives.

6.2 PINOs Limitations

6.2.1 Geometric Constraints

Most neural operator architectures assume regular grids or simple geometries, limiting their applicability to complex domains without significant preprocessing.

6.2.2 Interpolation vs. Extrapolation

PINOs perform well within the trained parameter range but may struggle with extrapolation beyond the training distribution, potentially leading to unphysical predictions.

7 Recent Advances and Future Directions

7.1 Causality and Temporal Modeling

Wang et al. [2022] introduced causal training methods for PINNs, ensuring that information propagates correctly in time-dependent problems. This addresses a fundamental limitation in standard PINN training where temporal causality can be violated.

7.2 Thermodynamic Consistency

Patel et al. [2022] developed thermodynamically consistent PINNs for hyperbolic systems, ensuring that the learned solutions respect conservation laws and entropy conditions. This work highlights the importance of incorporating physical principles beyond the governing PDEs.

7.3 Hybrid Approaches

Emerging research explores hybrid methods combining the strengths of both approaches:

- Using PINOs for coarse-scale modeling and PINNs for local refinement
- Pre-training with PINOs and fine-tuning with PINN-style physics constraints
- Multi-fidelity approaches combining fast operator predictions with high-accuracy PINN solutions

8 Implementation Considerations

8.1 Software Frameworks

Both PINNs and PINOs benefit from specialized software frameworks:

- **PINNs**: DeepXDE, NeuralPDE.jl, NVIDIA SimNet
- **PINOs**: FNO implementations in PyTorch, JAX-based neural operators

8.2 Hardware Requirements

PINNs typically require less memory but more computational time per training iteration due to derivative computations. PINOs require more memory for storing multiple parameter configurations but achieve faster training through vectorization.

9 Guidelines for Method Selection

Based on our analysis, we propose the following decision framework:

Choose PINNs when:

- Working with complex, irregular geometries
- Limited or no training data available
- High accuracy required for specific problem instances
- Solving inverse problems or parameter identification
- Physics constraints are critical and well-understood

Choose PINOs when:

- Conducting parametric studies or sensitivity analysis
- Real-time predictions required
- Multiple similar problems need to be solved
- Training data across parameter space is available
- Computational efficiency is prioritized over individual accuracy

10 Conclusion

PINNs and PINOs represent complementary approaches in physics-informed machine learning, each with distinct advantages and limitations. PINNs excel in scenarios requiring high accuracy for specific problems, particularly those involving complex geometries or inverse problems. Their ability to incorporate physical constraints directly into the learning process makes them invaluable for applications where physics-based regularization is crucial.

PINOs offer a paradigm shift toward operator learning, enabling efficient parametric studies and real-time applications. Their strength lies in learning mappings between function spaces, providing rapid inference across parameter ranges after initial training.

The choice between PINNs and PINOs should be guided by problem characteristics, accuracy requirements, computational constraints, and data availability. As the field continues to evolve, hybrid approaches combining the strengths of both methods show promise for addressing the limitations of individual approaches.

Future research directions include developing more robust training algorithms, extending operator learning to complex geometries, and creating unified frameworks that seamlessly integrate both paradigms based on problem requirements.

Acknowledgments

The authors acknowledge the contributions of the broader physics-informed machine learning community and the open-source software developers who have made implementations widely accessible.

References

Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. Physics-informed deep neural operators for learning effective equations in multiscale systems. *Journal of Computational Physics*, 464:111301, 2022. doi: 10.1016/j.jcp.2022.111301.

Amanda A Howard, Mauro Pereira, Jeet Pawan, Alejandro Strachan, George Em Karniadakis, and Wyatt Subber. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *Physical Review Research*, 4(2):023210, 2022. doi: 10.1103/PhysRevResearch.4.023210.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021. doi: 10.1016/j.jcp.2020.109951.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021. doi: 10.1038/s42254-021-00314-5.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. In *International Conference on Learning Representations*, 2021.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, Dule Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(1):1–27, 2023. doi: 10.1145/3648506.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Ravi G Patel, Indu Manickam, Nathaniel A Trask, Mitchell A Wood, Myoungkyu Lee, Ignacio Tomas, and Eric C Cyr. Thermodynamically consistent physics-informed neural networks for hyperbolic systems. *Journal of Computational Physics*, 449:110754, 2022. doi: 10.1016/j.jcp.2021.110754.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40): eabi8605, 2021. doi: 10.1126/sciadv.abi8605.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022.

Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021. doi: 10.1016/j.jcp.2020.109913.